Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _____
подпись инициалы, фамилия
«____» _____ 20 __ г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Интерактивная поддержка лекционных занятий и дистанционного обучения
тема

09.04.01 "Информатика и вычислительная техника"
код и наименование направления

09.04.01.04 "Технология разработки программного обеспечения"
код и наименование магистерской программы

Научный руководитель _____ профессор, д-р техн.наук А. И. Легалов
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Е. В. Ророт
подпись, дата инициалы, фамилия

Рецензент _____ заведующий кафедрой ИС, канд.пед.наук. С. А. Виденин
подпись, дата должность, ученая степень инициалы, фамилия

Нормоконтролер _____ В. И. Иванов
подпись, дата инициалы, фамилия

Красноярск 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Применение технических средств в ходе обучения стало неотъемлемым атрибутом образовательной деятельности. Основными составляющими учебного процесса являются получение знаний, навыков и компетенций, которые приобретаются путем применения различных форм обучения. Закрепление материала в таких дисциплинах как программирование обычно сопровождается решением задач после прочтения или прослушивания достаточно большого по объему теоретического материала. Этот теоретический материал обычно предоставляется в ходе лекционных занятий или во время самостоятельного прочтения соответствующих источников и зачастую не сопровождается достаточным числом примеров, необходимых для понимания. Во многом это происходит из-за нехватки времени на изложение всех возможных примеров, а также из-за того, что демонстрации подобных примеров без практической их реализации бывает недостаточным для получения нужных сведений.

Одним из возможных вариантов закрепления материала и понимания работы тех или иных конструкций языков программирования является непосредственное их использования в небольших демонстрационных программах, выдаваемых в качестве маленьких задач. Однако в большинстве случаев для оперативного решения подобных задач отсутствует техническая поддержка. Выполнение подобных задач на бумаге полностью убивает оперативность исполнения и контроля в виде обратной связи.

Зачастую процесс приобретения знаний является оторванным от их применения и освоения, что ведет к разрыву между получением информации и ее использованием. В результате чего к тому моменту, когда полученные знания нужно применить, информация о них уже забывается и ее приходится заново восстанавливать, пользуясь дополнительными источниками. Желательно, чтобы в процессе получения знаний можно было не только увидеть соответствующие примеры, но и непосредственно использовать их на

практике. Однако ряд существующих методов обучения в настоящее время не позволяют это сделать.

Необходимо ускорить процесс практического закрепления полученных знаний, увязав выполнение задач с непосредственным закреплением только что полученных теоретических знаний. Для этого необходимо иметь соответствующую инструментальную поддержку, позволяющую выполнять простые задания по программированию не только при выполнении практических заданий, но и на лекции.

Целью данной работы является разработка интерактивной системы обучения программированию, обеспечивающая написание программ студентами, и оперативным отображением текущих результатов выполнения в виде соответствующих статистических данных для преподавателя при непосредственном выполнении заданий во время изучения теоретического материала, которая могла бы применяться в ВУЗах для предоставления студентам легкого механизма закрепления теоретических знаний по программированию.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Сделать анализ особенностей процесса обучения для определения основных подходов к повышению уровню интерактивного взаимодействия между преподавателем и студентами во время проведения лекционных занятий, сформировать рекомендации по повышению интерактивности.

2. Разработать процессы, обеспечивающих интерактивное выполнение заданий и их обработку при проведении лекционных занятий.

3. Разработать архитектуру системы, поддерживающую интерактивное взаимодействие между преподавателем и студентами.

4. Реализовать систему интерактивного взаимодействия с применением современных информационных технологий.

В первой главе проводится анализ существующего лекционного процесса. Делается обзор функций существующих систем для обучения студентов

программированию. Описывается процесс проведения лекций с применением предполагаемых процессов интерактивного выполнения простых заданий. На основании этого формируются основные требования к разрабатываемой системе. Оценивается возможность ее реализации с использование существующих систем обучения.

Во второй главе представлены требования к разрабатываемому проекту и описаны пользовательские истории предполагаемого процесса обучения, который будет после внедрения разрабатываемой системы. На основе описанных пользовательских историй производится построение диаграмм прецедентов, которые отображают взаимодействие пользователей с возможностями системы и самой системы с пользователями.

Третья глава посвящена разработке внутренней структуре разрабатываемого проекта и интерфейса. На основе требований, описанных во второй главе была описана модель базы данных которая будет использоваться в системе, а также построены классы которые будут использоваться в системе. Описана программная реализация серверной и клиентской части.

В четвертой главе описано функционирование разработанной системы. Рассмотрено взаимодействие пользователей, выполняющих различные роли, с компонентами системы и между собой.

1 Анализ и особенности процесса обучения программированию

Вузовская система учебных дисциплин, которые относятся к информационным технологиям постоянно дополняются и совершенствуются.

Очевидно, что в системе подготовки студентов – программистов должно уделяться должное внимание теоретическим основам и конкретным технологиям алгоритмизации и программирования. Но в тоже время, уделяя много времени изучению только теории, будущие работодатели жалуются, что после окончания ВУЗа выпускник не может справляться с практической работой. В связи с этим должно уделяться должное внимание не только теории, но и практике.

1.1 Специфика преподавания программирования

Современный учебный процесс в ВУЗе представляет собой сочетание лекций, практических занятий, консультаций и самостоятельной работы студента с материалами. Но недавно результаты исследования, опубликованного в журнале Proceedings of the National Academy of Sciences [6] показали, что, студенты которые слушают обычные лекции, на 55% чаще заваливают экзамен, чем те, кто участвует в обсуждениях. На практиках в основном студенты показывают лабораторные работы, которые они выполняли дома и задают вопросы преподавателю, если что-то им не понятно. Студенты в основном показывают работы на своих собственных устройствах, так как не всегда есть возможность запустить программу на стационарных компьютерах которые находятся в аудиториях университета. Это в некотором смысле может являться проблемой для студентов, у которых отсутствуют переносные устройства. В тоже время для преподавателя — это рутинное занятие проверить лабораторные работы у студентов, практически на глаз, особенно данная рутина, может проявляться во время приближения сессии, в этот момент преподаватель даже может не успеть принять всех студентов.

1.2 Лекция – часть учебного процесса

Лекция – до недавних пор являлась ведущей формой обучения в ВУЗе. Ее основная дидактическая цель – формирование ориентировочной основы для последующего усвоения студентами учебного материала. Будучи главным звеном дидактического цикла обучения, она выполняет научные, воспитательные и мировоззренческие функции, вводит студента в творческую лабораторию лектора [7].

На смену слову в современном мире приходит ИТ, которое отражает новые процессы в дидактике, ведущими характеристиками профессии становятся компетенции, которые предполагают увеличение самостоятельной работы обучающего. На фоне всех этих изменений, возникает вопрос, а нужно ли использовать лекции, как форму обучения? Может быть, это является просто данью прошлого. Рассмотрим, какая сегодня роль лекции в обучении, и какой она должна быть в будущем.

О том нужна ли лекция, в современном образовании имеются различные точки зрения. Первая утверждает, что в век Интернета и при наличии большого количества книг в наше время, чтение лекций теряет смысл. Так как всю необходимую информацию можно найти либо в Интернете, либо в книгах. Без постоянного изучения книг и учебников, а также без владения практических навыков невозможно получить профессиональные знания. Другая точка зрения повествует о том, что, ни одна книга не сможет заменить живое общение.

Однако и та и другая мысль не имеют противоречий. Студент спокойно может найти необходимый ему материал, ставить вопросы, получать на них ответы, участвовать в каких-либо обсуждениях, что практически невозможно в традиционном обучении. При наличии книг и Интернета лекции стали просто не нужны, а преподаватели нужны только в роли консультантов.

Лекция может обеспечить усвоение темы на уровне ознакомлений. Именно в Европе был предложен (и аргументирован) переход на такой вид обучения, который бы включал только несколько ориентирующих лекций,

самостоятельную работу и семинарские занятий. В дальнейшем это стало основой, так называемой болонской системы обучения.

Можно выделить следующие аргументы против лекций, которые подтверждаются массовой практикой:

1. Лекции не стимулируют слушателя к мыслительной деятельности.

2. Информационных источников в современном мире настолько много, что необходимость лекций отпадает.

3. Лектор в большей степени навязывает свое мнение, из-за чего начинает «тормозить» мышление студентов.

4. Лекция как форма оторвана от аудитории и чаще служит для объемного охвата материала дисциплины.

5. Низкая активность слушателей, низкоэффективная обратная связь снижает эффективность усвоения учебного материала.

6. Чтение лекции некоторому среднему студенту (темы, содержание, методика и темп чтения традиционных лекций почти не зависят от качества восприятия и тем более усвоения материала студентами).

7. Высокие требования к мастерству лектора. Далеко не каждый преподаватель всегда эмоционально разнообразен и способен добиться высокого уровня внимания и активности слушателей на всем протяжении лекции.

8. Проведение лекционных занятий обычно жестко привязано к учебному расписанию. В случае отсутствия учащегося на занятиях ему бывает трудно восстановить пропущенный материал в полном объеме.

9. «Уплотнение» информации, на которое нередко идёт лектор, стремясь изложить весь программный материал в ограниченное время.

10. Пассивность студентов.

11. Сложность для лектора, организовать дальнейшую работу студентов над материалом лекции (лектор практически никак не организует последующую работу студентов во внеаудиторное время над разобранным материалом лекции, не направляет и не учит пользоваться литературой).

12. Однообразная форма изложения (со стороны большинства лекторов).

13. Несовременность излагаемого материала. Устаревшие образовательные программы. Студент не понимает перспективы полученных знаний, поэтому обучение не имеет для него личностного смысла.

14. Лектора можно заменить «телевизором», транслируя на нем записанную лекцию

Так же имеются и контраргументы представленных аргументов против лекций и их преимущества:

1. Стимулирование слушателей лекций можно реализовать при помощи интерактивности, закрепление теоретического материала на практике и введение в лекционный процесс соревновательных элементов.

2. Лектор может контролировать аудиторию и менять сценарий поведения в соответствии с реакцией аудитории, что позволяет более эффективно подавать материал. Непосредственная обратная связь обеспечивает более гибкую реакцию и ответы на вопросы.

3. Лекторы должны стараться повышать свой уровень проведения лекций и других занятий.

4. Преподаватель должен правильно планировать лекцию и целевую задачу, поставленную на лекции. Лекция не должна отменять другие виды деятельности студентов.

5. Лектор полностью контролирует содержание и последовательность подачи материала. Хорошие возможности во время лекции оперативно изменить последовательность и полноту раскрытия тем (или отдельных вопросов), темп изложения материала в зависимости от реакции слушателей, или от их уровня квалификации.

6. Возможность охвата большой аудитории.

7. Относительно низкие финансовые затраты на одного обучающегося (особенно при условии большого числа слушателей).

8. Эффективные лекции могут обеспечить побуждение к интеллектуальным открытиям путем представления сложных и провокационных идей [9].

9. Преподаватель может связать содержание лекции с реальными примерами из жизни, что делает полученные знания более значимыми.

Современное общество отходит от сферы неквалифицированного и малоквалифицированного труда, что определяет необходимость повышения профессиональной квалификации и переподготовки работников, их роста профессиональной мобильности. Как отмечает М.М. Акулич, акцент смещается с человека, знающего на человека, подготовленного к жизни [8]. Отсюда следует, что нынешнее общество предъявляет к образованию все новые и новые требования, и на этой основе разрабатываются новые стандарты с ориентацией на комплексный результат и переход от субъективно-объективных отношений между преподавателем и студентом к субъективно-субъективным.

Далее рассмотрим, из чего же состоит традиционная лекция, с двух сторон, со стороны преподавателя и со стороны студентов.

1.3 Традиционное лекционное занятие (описание процесса)

Главное в лекции — это мысль, логичность, умение показать интересное в излагаемом вопросе, дать формулировки — сжатые, точные и запоминающиеся, добиться подъема интеллектуальной энергии обучающихся, вызвать движение мысли вслед за мыслью лектора, добиться ответной мыслительной реакции. В этом случае будет обеспечено и непроизвольное запоминание. Лекция призвана вызывать у слушателей размышления, подсказывать направление самостоятельной работы мысли, побуждать к действию, быть школой научного мышления.

Рассмотрим, что значит лекция в современном мире. Лекция — это способ донести информацию до слушателей. Изобретена лекция была в древнем мире, когда книги как источник информации были чрезвычайно

дороги, и не вся полезная и новая информация в них записывалась. Но в современном мире информация стала легко доступной. Это и классические печатные книги, электронные книги, архивы научных статей. Практически на все вопросы можно получить ответ, для этого нужно просто «загуглить». На сегодня для лекций осталась одна ниша – это донесение новейшей, ещё не опубликованной информации из первых уст.

Таким образом, всё становится на свои места: преподаватель не является первоисточником информации; плюс заинтересованных лиц в аудитории, как вы уже знаете, не так много, иногда вообще нет. Вот и превращается лекция в абсурдный монолог на фоне студенческой болтовни. Далее мы подробно рассмотрим лекцию с двух сторон, со стороны преподавателя и со стороны студента.

1.3.1 Преподаватель

Преподаватель рассказывает материал. Демонстрация примеров проводится с применением доски или электронных средств обучения. Среди электронных средств можно отметить, что часть лекционного материала демонстрируется с использованием презентаций. Обычно с применением проектора. Помимо этого, используя проектор, преподаватель может демонстрировать примеры выполнения различных программ. Также, например, при демонстрации процесса разработки, он может набирать, запускать, компилировать, отлаживать различные программы. Все эти процессы отображаются на экране. Это значительно расширяет его возможности по сравнению с докомпьютерной эпохой, когда тексты программ писались на доске без какой-либо достоверной их проверки во время лекции. Правда, чаще всего, вместо непосредственного процесса создания программы, преподаватель предоставляет уже готовые исходные тексты, объясняя их функционирование. Возможно, что при этом используется набор слайдов, постепенно расширяющих текст описываемой программы, имитируя, тем самым процесс

разработки. Такой заранее подготовленный материал ускоряет процесс обучения, позволяя не засыпать студенту под монотонный стук клавиш или скрежет мела (скрип фломастера) по доске.

На традиционной лекции также возможен текущий контроль усвоения знаний, который выражается через следующие формы контроля.

1. Аудитории задаются устные вопросы. Отвечают отдельные студенты либо путем произвольного выбора, либо путем, кто первый успеет. Возможен вариант ответа с использованием голосования.

2. Выдается небольшое задание или тест для письменного выполнения, которое выполняется в течение нескольких (10-15) минут. После этого тесты собираются. Озвучиваются ответы. Окончательная проверка проходит значительно позднее. Ее результаты обычно доступны только на следующей лекции.

3. Ролевые игры (короткие), позволяющие закрепить преподаваемый материал через образные ассоциации.

4. Практическая проверка, такая проверка позволяет определить профессиональную готовность решать практические производственные задачи. Такая готовность определяется степенью сформированности системы умений и прежде всего профессиональных. Практическая проверка позволяет выявить, как студенты умеют применять полученные знания на практике, насколько они овладели необходимыми умениями, главными компонентами деятельности. В процессе выполнения профессиональных заданий студент обосновывает принятые решения, что позволяет установить уровень усвоения теоретических положений, т. е., одновременно с проверкой умений осуществляется проверка знаний. Проверка главным образом происходит, в основном, на практических занятий.

1.3.2 Студент

Сидит, слушает, смотрит. Некоторые делают пометки. Но сейчас, чаще всего, конспектирование не проводится, так как материалы лекций тиражируются и могут быть доступны в печатном или электронном виде. Выполняет тестовые и другие письменные задания после их выдачи. Отвечает на поставленные вопросы преподавателя. Зачастую отвлекается от процесса, так как существует множество отвлекающих факторов, такие как социальные сети, смартфоны, разговоры между студентами и т.д..

Сидя на лекции, часто очень сложно сконцентрироваться и слушать длительное время речь преподавателя. Поэтому студенты засыпают или достают гаджеты и тратят производительность своего электронного устройства на развлечения, общение или пролистывание новостных лент в социальных сетях. Студенты не чувствуют большой вовлеченности в процесс обучения на лекции.

Помимо всего этого массовая аудитория не позволяет лектору постоянно контролировать каждого студента. Поэтому отвлекшемуся студенту уже трудно возвращаться в процесс, так как часть материала, которой мог быть ключевым для понимания, оказывается не усвоенным.

В отличие от лекции, например, самостоятельный просмотр видеозаписи позволяет отмотать ролик назад и посмотреть его повторно, если что-то не понятно. Если это делается в индивидуальном порядке, то студент может сам выбирать время, темп, повторы, моменты, когда можно отвлечься или снова приступить к процессу изучения.

1.3.3 Самостоятельная работа студента

Новая парадигма современного образования определяет смену приоритетов, самостоятельная работа становится не просто одним из методов образовательного процесса, а фундаментом, который формирует

профессиональную самостоятельность студента, способствующую более эффективному овладению учебными материалами, профессиональному интересу.

Самостоятельная работа подразделяется на несколько видов:

1. Самостоятельна работа в ВУЗе (на лекции, на практических занятиях)
2. Самостоятельная работа вне ВУЗа (домашние задания)

1.4 Интерактивный подход к лекциям

Рассмотрев выше перечисленное, можно прийти к выводу, что лекции в ВУЗе нужны, но они нужны в каком-то изменённом виде, например, проводить лекции в виде дискуссий, проектов. Так как традиционные лекции уже не приносят много пользы для студентов. То есть подразумевается, что вместо того чтобы давать на лекциях «сухой» теоретический материал, занятия можно перевести в разряд интерактивных. В этом случае предполагается, что будет идти предварительная подготовка группами, на которые будут разбиты студенты, это так же позволит их подготовить в будущем для командной работы. В отличии от практических и лабораторных занятий в том, что, это будет происходить при больших аудиториях. Каждая группа может прорабатывать альтернативы того или иного вопроса, что будет позволять инициировать дискуссии. Аналогично выше описанной схеме на лекции можно будет проводить разнообразные викторины, мини проекты и т.д.

Исходя из выше сказанного получается, что студент должен приходить на занятия уже подготовленным. Т.е. получается, что предварительно студент дома должен усвоить какой-то материал. По сути, получается, самостоятельно изучить лекцию. Это все делается для того чтобы студент активно участвовал в интерактивном процессе.

Далее рассмотрим подробно плюсы традиционных лекций, а также набирающее популярность дистанционных, для того чтобы попытаться составить «формулу» хорошей лекции в ВУЗе.

Таблица 1 – Сравнение лекций и дистанционного обучения

Традиционная лекция	Дистанционное обучение
Лекция дает возможность во время лекции оперативно изменить последовательность и полноту раскрытия тем (или отдельных вопросов), темп изложения материала в зависимости от реакции слушателей, или от их уровня квалификации.	Студент, обучающийся дистанционно, решает самостоятельно, когда и сколько времени ему уделять на изучение материала. Он строит для себя индивидуальный график обучения.
Преподаватель может связать содержание лекции с реальными примерами из жизни, что делает полученные знания более значимыми. При этом постоянно обновляемые или дополняемые материалы позволяют насытить изучаемую дисциплину теми же самыми реальными примерами из жизни. Но как говорится это уже не из «первых рук».	Учащимся дистанционно не нужно беспокоиться о том, что они отстанут от своих однокурсников. Всегда можно вернуться к изучению более сложных вопросов, несколько раз пересмотреть, перемотать видео-лекции, если что-то не понятно в данной теме, а уже известные темы можно пропустить.

Продолжение таблицы 1 – Сравнение лекций и дистанционного обучения

Традиционная лекция	Дистанционное обучение
При проведении традиционной лекции происходит личный контакт студентов друг с другом и преподавателем, что позволяет развивать коммуникабельность и навыки работы в команде.	Студенты могут учиться, не выходя из дома или офиса, находясь в любой точке мира. Чтобы приступить к обучению, необходимо иметь компьютер с доступом в Интернет.
Традиционное обучение позволяет проследить за тем, честно и самостоятельно ли студент выполняет задания.	Как показывают исследования американских ученых, результаты дистанционного обучения не уступают или даже превосходят результаты традиционных форм обучения. Большую часть учебного материала изучают самостоятельно. Это улучшает запоминание и понимание пройденных тем. А возможность сразу применить знания на практике, на работе помогает закрепить их.
Позволяет мотивировать студентов к обучению.	Связь с преподавателями, осуществляется разными способами: как on-line, так и off-line. Проконсультироваться с преподавателем с помощью электронной почты иногда эффективнее и быстрее, чем назначить личную встречу при очном обучении.

Окончание таблицы 1 – Сравнение лекций и дистанционного обучения

Традиционная лекция	Дистанционное обучение
Преподнесение более нового современного материала по изучаемой теме. Не всегда все самое новое успевает публиковаться в общем доступе.	Промежуточная аттестация студентов дистанционных курсов проходит в форме on-line тестов. Поэтому у учащихся меньше поводов для волнения перед встречей с преподавателями на зачетах и экзаменах. Исключается возможность субъективной оценки: на систему, проверяющую правильность ответов на вопросы теста. При этом обычно итоговый контроль часто осуществляется в рамках независимого аудита. Где нельзя списать. Поэтому готовиться к сдаче необходимо серьезно.
	При традиционном обучении преподавателю довольно трудно уделить необходимое количество внимания всем студентам группы, подстроиться под темп работы каждого. Использование дистанционных технологий подходит для организации индивидуального подхода. Кроме того, что учащийся сам выбирает себе темп обучения, он может оперативно получить у преподавателя ответы на возникающие вопросы.

Методы повышения интерактивности:

Исходя из аргументов против лекций попробуем сформулировать идеи как можно это исправить.

1. Стимулирование студентов к мыслительной деятельности:

Лектор должен не просто делать подачу материала, сухой текст, монолог. Но выслушивать мнения студентов, по каким-либо вопросам касаемо изучаемого материала, в виде каких-либо опросов, вопросов или какими-либо другими методами.

Наверное, возможность взаимодействия – это самый существенный фактор. Он может формироваться за счет проектного обучения. Однако в этом случае встает вопрос: лекции или практические занятия?

2. Лектор должен идти в ногу со временем:

Лектор должен стараться следить за новинками в области своей дисциплины. Изучать современные веяния по преподаваемому им предмету.

При этом может быть, после усвоения нового материала имеет смысл выкладывать его в дистанционную среду. Это будет напоминать то, что сейчас делается блогерами, оперативно пишущими тексты или выкладывающими видео.

3. Выслушивание мнений студентов, обратная связь:

Получение обратной связи от студентов о проведенной лекции также возможен в устной форме, письменной форме либо с использованием сторонних программ.

4. Смотреть, как усваивает материал каждый студент (контроль):

Контроль усвоения материалов лекции. Такой контроль возможен как через проведение лабораторных работ, диалогов со студентами по данной теме в виде вопросов и ответов. Также возможен вариант контроля с использованием современной техники и различных программ. Однако лучше всего студент работает непосредственно в присутствии преподавателя. Т.е. преподаватель молчит и наблюдает и в случае чего подсказывает индивидуально.

5. Организация работы студентов после прослушивания лекций:

Выдача, каких-либо заданий после лекций и их выполнение студентами. Это можно делать дистанционно.

6. Нужно предусмотреть возможность возвращаться к уже изученному материалу. Это можно решить несколькими способами:

а) Конспектирование лекций студентами, что в современных реалиях будет тормозить процесс обучения;

б) Выкладывать в онлайн текстовый вариант лекций и презентации;

в) Видеозапись лекций и также выкладывание их в онлайн;

Третий вариант наиболее полезен, а особенно если он будет вкупе со вторым.

7. Изучение некоторого материала самостоятельно, но для этого необходимо предоставление того самого материала. Закрепление его на практике. Близко связанно с первым пунктом, из этого следует, что нужно подготавливать материал в удобном виде.

8. Связь с преподавателем должна быть не только off-line, но и on-line, это будет позволять не держать и не запоминать этот вопрос до следующей встречи с преподавателем. Методы решения:

А) Электронная почта

Б) Skype

В) Чаты

9. Предоставление некоторых аттестаций в виде on-line тестов, а не только устно.

10. Контроль усвоения материала студентами – этот вопрос также решается либо с помощью тех же тестов или заданий на лекции, либо при помощи практических занятий и выполнение студентами лабораторных работ.

11. Стимулирование студентов к мыслительной деятельности:

Это можно решить с помощью предоставления небольших тестов или задач прямо на лекции, что позволит студентам вернуться в рабочий процесс.

12. Контролирование успеваемости студентов. Выставление оценок не только по результатом экзамена.

Далее из всего выше сказанного, попробуем описать, как на практике можно изменить проведение занятий (лекций), рассматривать мы это будем на примере предмета «технологии программирования».

Учебный процесс должен быть организован так, чтобы практически все студенты были вовлечены в процесс познания, имели возможность понимать и размышлять по поводу того, что они знают и думают.

1.5 Методы закрепления теоретических знаний

1.5.1 Лабораторные работы

На мой взгляд, как минимум процесс приема лабораторных работ может быть автоматизирован. Что позволит не тратить время на прием работ, а тратить это время, например, на объяснение каких-то непонятных моментов и ответы на вопросы студентов.

Конечно, лабораторные работы можно, например, отправлять и по электронной почте. Так даже позволяют делать не которые преподаватели, но как по мне это не сильно то и экономит время преподавателя, ему приходится постоянно проверять свою почту и в тоже время он может просто пропустить письмо о студента и не проверить. Так же преподавателю придется, если это какая-то программа, запускать на своем компьютере, что так же будет тратиться время, причем не маленькое.

Данный процесс можно и даже нужно автоматизировать, например, используя какие-либо системы. В которые студенты могут оправлять свои работы, а уже преподаватель будет смотреть предоставленные работы в этой системе. В Институте Космических и Информационных технологий СФУ используется система Moodle, в которую студенты могут отправлять свои ответы и работы, и в тоже время она не позволяет взять просто и посмотреть

эту работу, особенно если это программа и ее нужно запустить чтобы удостовериться, что она работает и при том правильно. Поэтому данная система не подходит под такие нужды. Например, для таких нужд в Гарвардском университете используется система CS50 основанная на Cloud9, она имеет собственную IDE, где студенты пишут свои программы и к которым тут же имеет доступ преподаватели и их ассистенты. Преподаватели тут же могут посмотреть и оценить работу студента или же если у студента что-то не получается, то могут просто посмотреть на его код программы и помочь если требуется.

1.5.2 Лекции

В процесс проведения лекций можно внести элементы интерактивности и обратной связи.

Во время объяснения материла на лекции, преподаватель может делать паузы в рассказе и давать небольшие задачи студентам для закрепления теоретического материала. На данный момент такое тоже происходит на лекциях, но реализуется это с помощью написания таких небольших задачек на листочках, которые преподавателю еще нужно будет проверить после завершения лекции. И только на следующем занятии объявить студентам полученные оценки.

Процесс выдачи и проверки таких заданий можно автоматизировать, что так же внесет интерактивность в процесс проведения лекционных заданий. Автоматизация поможет быстрее проверять выполненные задания и вносить какие-либо коррективы в процесс проведения лекции, например, если какие-то моменты не понятны, то еще раз их объяснить.

1.6 Обзор существующих систем обучения

1.6.1 CS50 IDE

CS50 IDE [9] - представляет собой облачную среду разработки, основанную на базе Cloud9 [2], которая в свою очередь работает на облачной среде Ubuntu. Среда используется для обучения студентов Гарварда программированию. IDE имеет редактор, к которому получают доступ из браузера, он поддерживает подсветку синтаксиса и завершения слов, имеется отладчик на основе графического интерфейса. При доступе к среде разработки пользователь получает контроль над облачной средой Ubuntu.

1.6.2 Moodle

Moodle (модульная объектно-ориентированная динамическая учебная среда) — это свободная система управления обучением, ориентированная прежде всего на организацию взаимодействия между преподавателем и студентами, хотя подходит и для организации традиционных дистанционных курсов, а так же поддержки очного обучения.

Используя Moodle преподаватель может создавать курсы, наполняя их содержимым в виде текстов, вспомогательных файлов, презентаций, опросников и т.п. Для использования Moodle достаточно иметь компьютер с доступом в интернет и браузер на этом компьютере, что делает использование этой учебной среды удобной как для преподавателя, так и для студентов. По результатам выполнения учениками заданий, преподаватель может выставлять оценки и давать комментарии. Таким образом Moodle является и центром создания учебного материала и обеспечения интерактивного взаимодействия между участниками учебного процесса.

1.7 Обзор open source облачных IDE, на основе которой можно реализовывать свою систему

1.7.1 Cloud 9

Cloud9 – интегрированная среда разработки, предоставляемая по модели облачных вычислений, была запущена 28 февраля 2011 года, разработчиками текстового редактора Ace [10]. Как указывают авторы на сайте [11] поддерживает около 40 языков программирования, таких как JavaScript, PHP, Python, C++ и др. IDE полностью написана на JavaScript и использует на стороне сервера Node.JS. В качестве текстового редактора используется Ace.

Авторы описывают так, сервис предназначен для всех, кто занимается командной разработкой. В браузере находится редактор кода, среда для эмуляции и различные инструменты. Браузер делает процесс разработки не привязанным, к какой-то определенной операционной системе. Готовый программный код можно разворачивать прямо из браузера.

Плюсы:

1. Редактор разработан на открытых технологиях, из-за чего поддерживает модули на основе открытого исходного кода, а также плагины и расширения новой функциональности.

2. Совместная работа над кодом. В cloud9 доступен просмотр, редактирование, комментирование и обсуждение кода и различных версий разрабатываемого ПО прямо из браузера. Поддерживается интеграция с bitbacket и github. Все изменения и комментарии отображаются в режиме реального времени. Данная особенность хорошо не только для командной разработки, но и для проведения различных занятий, лекций и вебинаров, когда имеется необходимость комментирования для широкой аудитории написанного кода.

3. Работа на облачных технологиях. Все инструменты расположены в облаке. Имеется возможность запускать в облаке разрабатываемые

приложения. Будь то это какие-то веб-приложения, написанные на JS, либо C++ программы.

4. Имеется возможность клонирования рабочей среды. С клонированием рабочей среды, делается полная копия кода, настроек среды и расположения вкладок.

5. Отладчик, позволяет устанавливать точки остановки, проверять переменные любого приложения. Выполнение кода в сеансе отладки.

6. Cloud9 IDE обеспечивает различные уровни интеллектуального и гибкого автозаполнения кода. Автозаполнение основано не только на содержимом файлов, но также на стандартных функциях и инструментах языка.

7. История изменения файлов. Возможность просмотра файла во временной шкале, то, как выглядел файл в определенный момент времени. Если нажать кнопку воспроизведения как именно вводился текст в файл, что будет очень полезно при объяснении студентам написанного кода.

1.7.2 Codeboard

Codeboard [12] – представляет собой веб – IDE для обучения программированию на занятиях. Позволяет создавать упражнения в многофункциональной среде IDE и легко делится ими, используя их url – адреса. Поддерживает языки программирования: C, C++, Eiffel, Haskell, Java, Python.

Плюсы:

1. Имеет автоматическую классификацию документов на основе модульных тестов или индивидуальных тестовых драйверов.

2. Присутствует возможность проверять предоставленные документы непосредственно в Codeboard.

3. Имеется возможность следить за прогрессом студентов, используя при этом конкретные статические данные проектов студентов.

4. Codeboard поддреживет LTI v1.1, протокол взаимодействия IMS. LTI позволяет интеграцию образовательных платформ с внешними инструментами. Образовательные платформы могут взаимодействовать с Codeboard через LTI. Образовательная платформа берет на себя роль потребителя инструментов, а Codeboard выступает в качестве поставщика инструментов.

5. Возможно, делится проектами с другими пользователями, предоставляя им url – адрес проекта.

6. Проверка заданий студентов происходит на тестах.

1.7.3 Codebox

Codebox – интегрированная среда разработки построенная на облачных вычислениях. Codebox поставляется в двух версиях, в виде веб-клиента, либо плагина для браузера Google Chrome. Codebox полностью разработан по принципам open source, весь исходный код доступен на github.

Плюсы:

1. Совместная работа, несколько разработчиков могут работать над одним приложением одновременно, построена на принципах Google Docs.

2. Обучение программированию. Для преподавателя имеются возможности совместной работы, что позволяет проводить наглядные уроки, проверять работы студентов и править их.

3. Практически любой код можно запустить прямо в браузере. Для тестирования и отладки имеется полноценный терминал.

4. Доступ offline, если в какой-то момент нет доступа к сети Интернет, то имеется плагин для браузера Google Chrome, который позволяет разрабатывать и без интернета.

1.7.4 Итоги выбора

На основании выше сказанного логично предположить, что Codeboard подходит больше для обучения программированию, но только начальному, в то время как cloud9 поддерживает разработку крупных проектов, что будет полезно на старших курсах.

Cloud9 поддерживает разработку плагинов, что нам в конечном итоге будет полезно, если мы будем разрабатывать АРМ для студентов на его основе. Но в тоже время в Codeboard изначально поддерживает функционал, позволяющий управлять и оценивать студентов, данный функционал основан на тестах.

Так как cloud9 не поддерживает выдачи определенных задач, автоматическую проверку программ на тестах, для этого придется писать определенные плагины, разбираться в архитектуре данной системы, что может занять определенное время, решено не использовать данную систему.

У codeboard отсутствует возможность массово выдавать задачи студентам одним разом, что в итоге может привести к временным потерям при проведении лекций, когда преподаватель должен иметь возможность за один раз дать одну и ту же задачу, большому количеству студентов.

Из – за таких казалось бы небольших ограничений этих систем, было принято решение разрабатывать свою собственную, под свои нужды. В тоже время разработка собственной системы позволит, безболезненно её расширять по мере необходимости.

2 Разработка требований к разрабатываемой системе

2.1 Пользовательские истории предполагаемого процесса обучения

В данном разделе описываются требования к разрабатываемой системе по средствам описания пользовательских историй (user story).

Пользовательские истории описываются со стороны пользователей, т.е. студента и преподавателя.

2.1.1 Студент. Работа на занятиях

Приходит на лекционные занятия. Используя лабораторный десктоп, или собственный ноутбук, или планшет, или смартфон и т. д. входит на страницу системы, где отображается информация о проводимом занятии, где можно посмотреть информацию, презентацию по данной теме и ожидает информации о получении очередного задания. В какой-то момент времени для него открывается задание. Студент нажимает кнопку «Начать выполнение» и входит в среду, в которой описано задание и открыт экран для набора и редактирования исходного текста программы. Помимо этого, появляется рабочий каталог с возможным списком предопределенных файлов и файлом проекта, описывающим формирование требуемого загрузочного модуля. Часть файлов может быть уже заполнена полностью или частично (в случае использования некоторого шаблона частичное заполнение позволяет сократить набор текста программы). Редактируя представленный шаблон, студент формирует окончательный код требуемого задания, возможно в несколько итераций, связанных с компиляцией и запуском. Взаимодействие с системой при этом осуществляется с использованием консоли, обеспечивающей удаленный доступ к системе поддержки на сервере. После того, как студент удостоверится в корректном выполнении программы, он может нажать на кнопку: «Задание выполнено». После чего система переходит в режим

ожидания дальнейших инструкций от преподавателя. При этом вполне возможно, что задание выполнено некорректно. Это неважно с точки зрения формирования обратной связи. Дальнейшие действия определяются преподавателем в соответствии с особенностями проводимого занятия.

В рамках дальнейших инструкций студент может получить от системы информацию (команды, сообщения), оповещающие о дальнейшем взаимодействии с системой или предоставляющие различные сведения (результат выполнения задания, дальнейшие инструкции). Эта информация может инициироваться как системой, так и преподавателем в зависимости от режима работы (аудиторная, удаленная; с преподавателем, с системой электронного обучения).

2.1.2 Запуск написанных программ

Запуск разработанной программы состоит из двух частей, это компиляция и собственно запуск. Запуск и компилирования программы производится через пункты меню.

2.1.3 Предварительная настройка системы перед началом работы

1. Когда пользователь первый раз входит в систему, ему предлагается ее настроить. Также настроить систему под себя возможно и в дальнейшем.

2. Настройка включает в себя выбор языка программирования, цветовой схемы IDE, компилятора или интерпретатора и тд.

2.1.4 Преподаватель

Взаимодействует с системой через рабочее место преподавателя. В ходе занятий выбирает дисциплину, тему занятия. На текущем экране отображается список заданий. Каждое задание предварительно настроено (в диалоговом окне)

на определенный режим выполнения (например, использование таймера с указанием времени; задание одно для всех или на группу, или на каждого студента...). Режим задания может корректироваться путем перехода в него по нажатию кнопки «Редактировать». Невозможно удаление заданий в режиме преподавателя. В ходе занятия преподаватель выбирает задание из списка и запускает его кнопкой «Выполнить». Задание уходит на выполнение. В интерфейсе преподавателя запускается таймер обратного отсчета при его установке. Таймер может быть установлен заранее при создании задания или его редактировании. Возможна коррекция значения непосредственно перед запуском задания на выполнение. Помимо этого, отображаются кнопки «Прервать выполнение задания», «Продлить выполнение задания». В последнем случае можно указать дополнительное время, которое изменяет значение таймера обратного отсчета. После прерывания задание можно также снова продлить или нажать на кнопку «Завершить».

После того как студенты выполнили задания, преподаватель может проверить правильность их выполнения. Для этого преподаватель открывает список студентов, выбирает того студента задания, которого он хочет проверить. После того как выбор сделан, преподавателю открывается код программы, который преподаватель может запустить также, как и студент запустить в двух режимах. Преподаватель имеет также право редактировать код студента или оставлять какие-то комментарии по коду или по программе в целом, которые в дальнейшем увидит студент этой программы. При этом встроенная система поддержки версий сохраняет версию преподавателя отдельно от версии студента. Это чтобы можно было фиксировать портфолио и взаимодействие. Также система выдает общую статистику. Количество выполненных заданий (например, по прохождению всех контрольных тестов), активность студентов (по числу обращений, фиксации интенсивности набора и т. д.).

2.2 Обзор диаграмм прецедентов

На основе описанных в пункте 2.1 пользовательских историях были построены диаграммы прецедентов.

В описанной предметной области выделяется два следующих актера.

1. Актер *Студент,* использует систему для выполнения различных заданий, путем написания программного кода и его последующей отладкой.

2. Актер *Преподаватель,* использует систему, для выдачи заданий студентам, контроля их выполнения и дальнейшей их проверки. Может также писать программный код, для дальнейшей демонстрации.

На основе вышеизложенного можно выделить следующие прецеденты, для каждого актера по отдельности:

Таблица 2 – Описание прецедентов актёра «Студент»

Прецедент	Краткое описание
Вход в систему	Запускается студентом. Позволяет авторизовать пользователя по логину и паролю.
Информация о задании	Запускается преподавателем. Позволяет просматривать информацию о задании, которое требуется выполнить.
Выполнение задания	Запускается студентом. Позволяет писать, редактировать программный код.
Написание программы	Запускается студентом. Позволяет редактировать, программный код, создавать, удалять файлы с программным кодом.
Компиляция и сборка	Запускается студентом. Позволяет скомпилировать или собрать программу, для того чтобы проверить ее работоспособность.

Окончание таблицы 2 – Описание прецедентов актёра «Студент»

Прецедент	Краткое описание
Отладка программы	Запускается студентом. Позволяет отладить программу, посредством постановки точек остановки, и различных данных которые предоставляет программа, не посредственно в какой-то момент ее выполнения.
Завершение задания.	Запускается студентом или преподавателем. Позволяет завершить выполнение задания, и посмотреть результаты выполнения задания.

Таблица 3. – Описание прецедентов актёра «Преподаватель»

Прецедент	Краткое описание
Вход в систему	Запускается преподавателем. Позволяет авторизовать пользователя по логину и паролю.
Создание / редактирование темы	Запускается преподавателем. Позволяет создавать и редактировать темы занятий, добавлять в них описание, и различный материал, например, в виде презентаций.
Создание / редактирование задания	Запускается преподавателем. Позволяет добавлять различные задания которые относятся к определенным занятиям.
Выбор темы	Запускается преподавателем. Позволяет выбрать тему занятия. В результате система отображает информацию о теме занятия в интерфейсе студента.
Назначение задания	Запускается преподавателем. Позволяет назначить то или иное задание студентам.

Окончание таблицы 3 – Описание прецедентов актёра «Преподаватель»

Прецедент	Краткое описание
Завершить задание	Запускается преподавателем. Позволяет автоматически завершить выполнение задания студентами.
Посмотреть результаты	Запускается преподавателем. Позволяет посмотреть результаты выполнения заданий студентами.

Диаграмма пользователей представлена на рисунке А.1 приложения А.

2.3 Поток событий для прецедентов диаграммы

Поток событий для прецедента «Информация о задании»

- **Главный поток**

Прецедент начинает выполнятся, когда студент подключатся к системе и вводит свой логин и пароль. Система проверяет правильность пароля и выводит возможные варианты действия: начать выполнение, посмотреть задавание, отмена.

- **Под потоки**

Посмотреть задание: система отображает диалоговое окно, содержащее текст условия выполнения задания. Пользователь выбирает начать выполнение задания или отменить.

Начать выполнение: система отображает IDE, где находится список файлов, в которых пользователь будет писать код.

Поток событий для прецедента «Выполнение задания»

- **Предусловия**

Перед прецедентом «Выполнение задания», должны быть выполнены прецеденты «Выбор задания» и «Информация о задании».

- **Главный поток**

Прецедент начинает выполнятся, когда студент подключатся к системе и вводит свой логин и пароль. Система проверяет правильность пароля и выводит возможные варианты действия: написание программы, компиляция и сборка, отладка программы.

- **Под потоки**

Написание программы: система отображает тестовый редактор, содержащее текст написанной программы, файловый менеджер, где пользователь выбирает файлы которые он будет редактировать. Консоль, где пользователь может вводить различные команды, которые могут запускать следующие под-потоки, компиляция и сборка и отладка программы.

Компиляция и сборка: система запускает компиляцию и сборку написанной пользователем программы и выводит результат выполнения программы. Компиляцию и сборку программ можно производить несколькими путями, запуск компиляции и сборки путем ввода команд в окно терминала и путем выбора пунктов меню.

Отладка программы: система компилирует программу и позволяет проводить отладку написанной программы путем ввода команд в окно терминала, запуска отладчика из пункта меню и расстановки точек остановки в текстовом редакторе.

Поток событий для прецедента «Завершение задания»

- **Главный поток**

Прецедент начинает выполнятся, когда студент выбирает в пункт меню «завершить задание». выполненное задание сохраняется на сервере, а так же на сервере сохраняется вся статистика по выполнению, такая как время выполнения и тд..

Поток событий для прецедента «Выбор темы»

- **Главный поток**

Прецедент начинает выполнятся, когда преподаватель подключатся к системе и ввел свой логин и пароль. Система проверяет правильность пароля и выводит возможные варианты действия: Выбрать тему занятия, отмена.

- **Под-потоки**

Выбор темы занятия: Система отображает список из доступных для выбора тем занятий. Преподаватель должен выбрать одну из предложенных тем занятий.

Поток событий для прецедента «Назначение задания»

- **Предусловия**

Прецедент начинает выполняться только после выполнения предыдущего прецедента «Выбор темы занятия».

- **Главный поток**

Открывается список доступных заданий для выбранной темы занятия. После открытия списка задания имеется два варианта развития событий. Выбор задания для всей группы и выбор задания для определенного студента. После выбора задания открывается окно в котором преподаватель запускает выполнение. После запуска выполнения запускается таймер, который показывает сколько времени прошло с момента начало выполнения задания.

Поток событий для прецедента «Завершить задания»

- **Предусловия**

Перед выполнением данного прецедента, должен быть выполнен предыдущий прецедент «Назначение задания».

- **Главный поток**

В окне, открытом в предыдущем прецеденте рядом с таймером доступна кнопка завершить задание. По нажатию на которую таймер останавливается и у

всех студентов которым преподаватель назначил данное задание, оно завершается и сохраняются результаты выполнения задания.

Поток событий для прецедента «Посмотреть результаты»

- **Главный поток**

Преподаватель выбирает в пункте меню посмотреть результаты. Открывается окно в котором отображается статистика по выполненным заданиям студентами. В виде таблицы отображается список студентов, по нажатию на которого можно посмотреть индивидуальную статистику данного студента.

Диаграмма деятельности преподавателя представлена на рисунке А.2 приложения А.

3 Разработка архитектуры и основных технических решений

3.1 Структура системы

На рисунке 3.1 изображена диаграмма развертывания, из нее видно, что разработанная система состоит из 3 частей, базы данных, сервера приложения и клиентов.

Более подробная структурная схема представлена на рисунке 3.2. Из нее видно, что каждый элемент системы состоит из блоков, между которыми передаются данные.

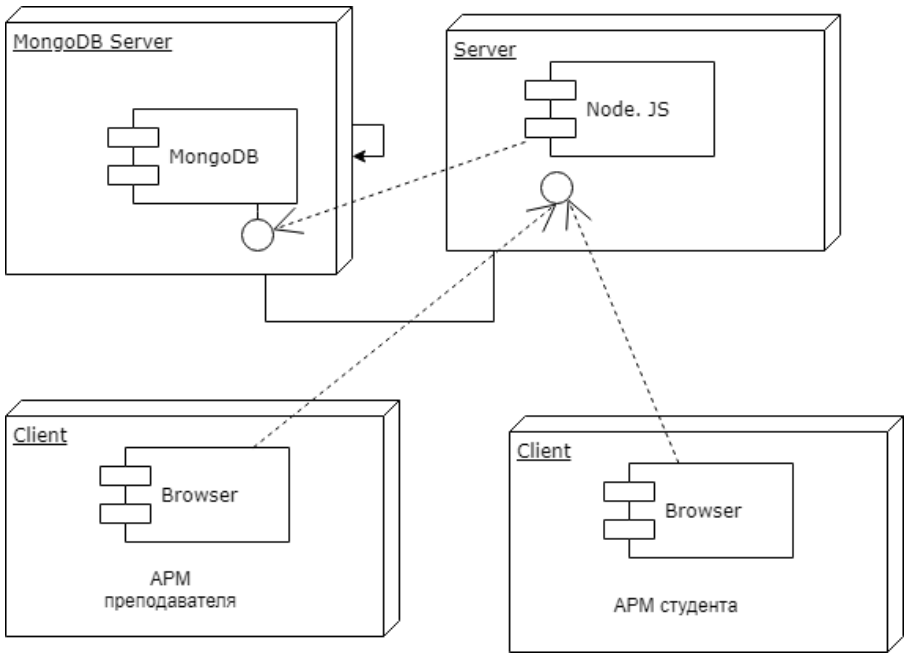


Рисунок 3.1 – Диаграмма развертывания

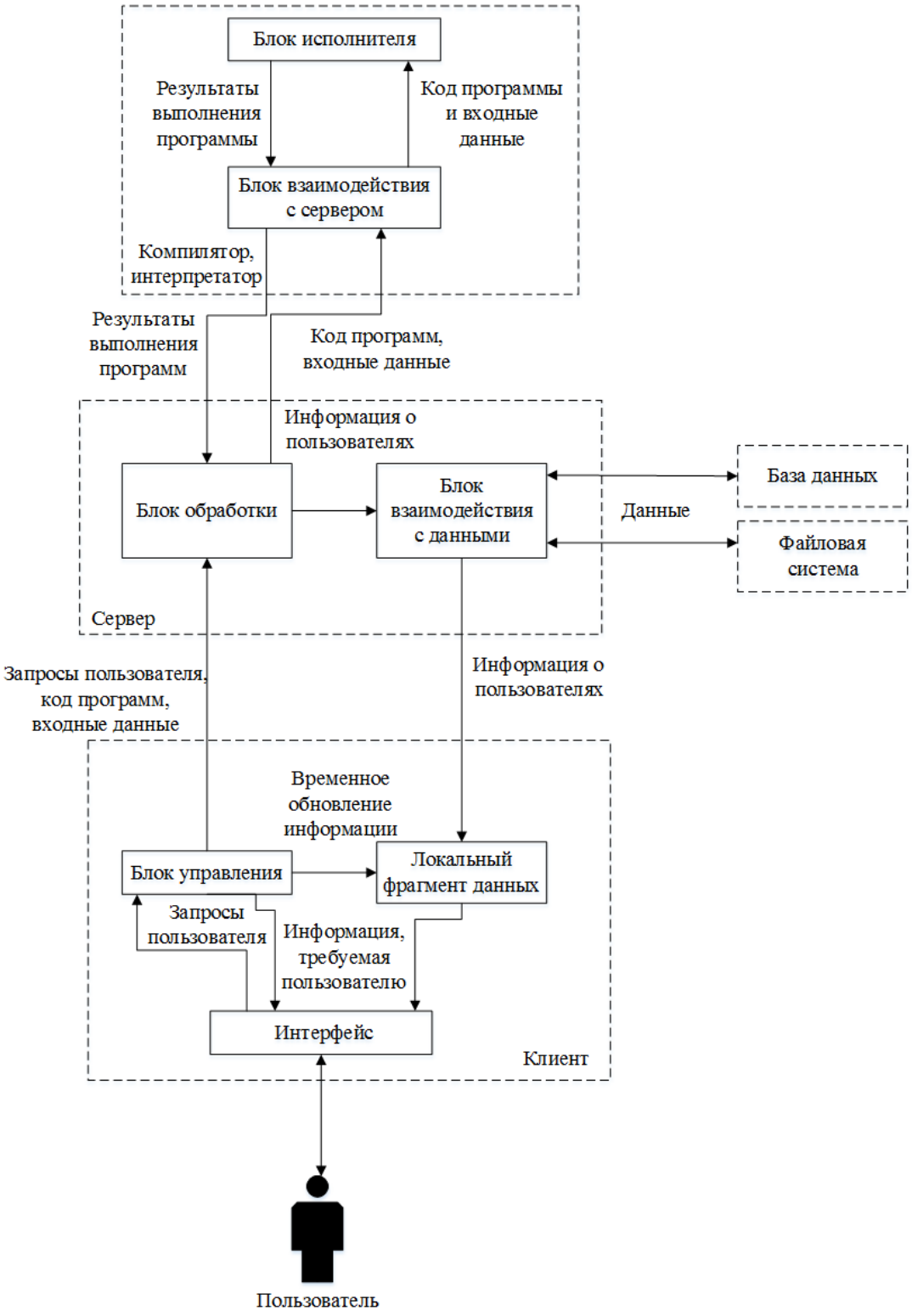


Рисунок 3.2 – Структурная схема

Пользователь с помощью графического интерфейса запрашивает требуемую ему информацию, например, профиль пользователя, код программы. Сервер обрабатывает запрос и записывает результат, в зависимости от данных либо в базу данных, либо в файловую систему. Далее результат сохраняется в

локальном фрагменте данных и отображается через графический интерфейс пользователя. Если результат уже был сохранен на клиенте, то повторно запроса к серверу не будет происходить, и данные будут браться напрямую из локального фрагмента данных.

Если пользователь запрашивает исполнение программы, то сервер будет передавать исходный код и если это необходимо, то какие-то данные, которые требуются для компилятора или интерпретатора. Далее результат возвращается обратно на сервер и после передается клиенту для отображения в графическом интерфейсе.

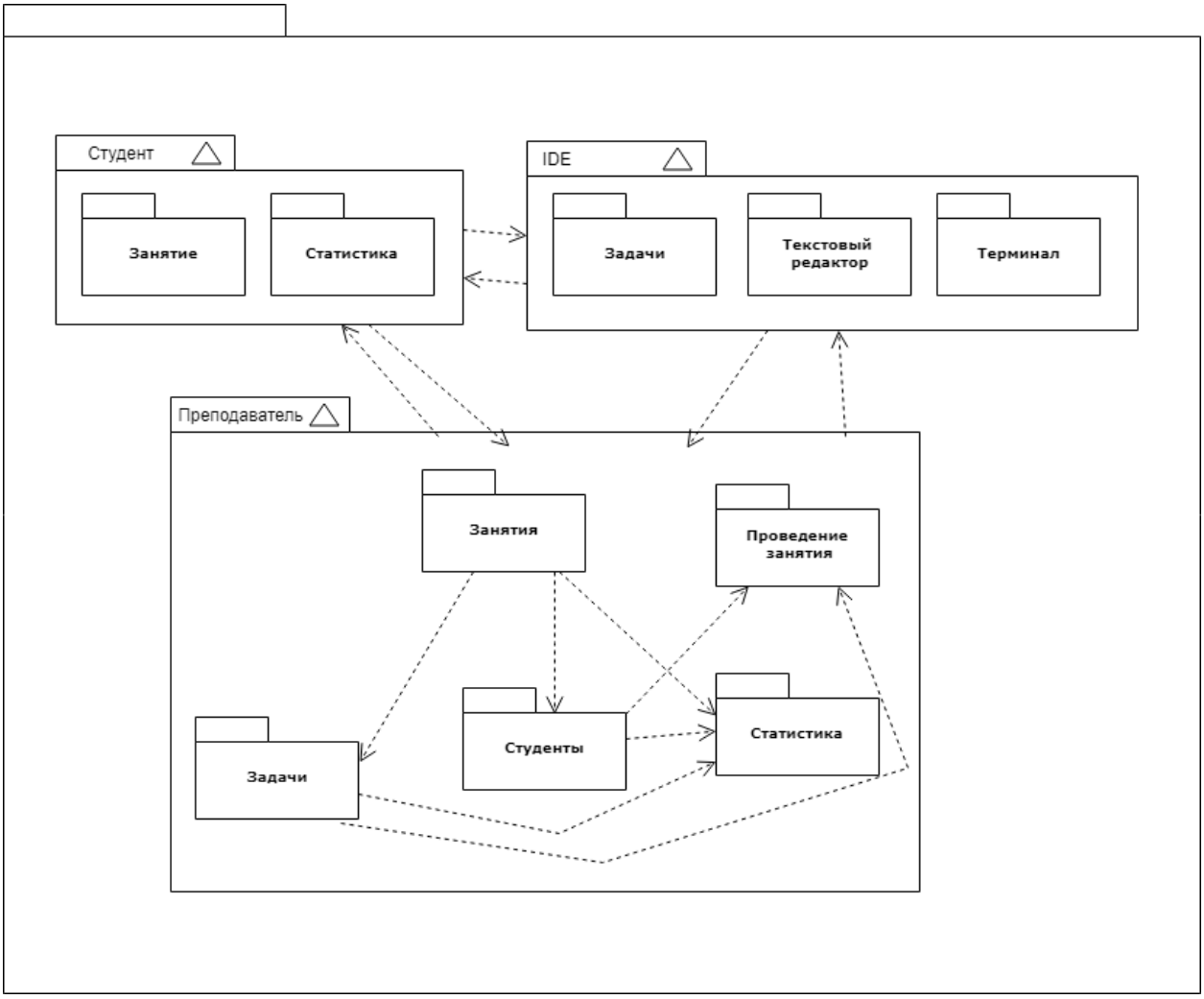


Рисунок 3.3 – Диаграмма модулей

Вся система состоит из 3-х больших модулей (рисунок 3.3), которые в свою очередь состоят из модулей меньше.

Модуль преподаватель отвечает за взаимодействие с системой преподавателей. Этот модуль состоит из 5-ти подмодулей. Занятия, отвечает за создание и редактирование занятий и наполнение их различными материалами. Модуль задачи, отвечает за создание задач и наполнение их тестовыми данными. Проведение занятий, позволяет проводить занятия со студентами и выдавать им различные задачи во время проведения и контролировать ход их выполнения. Модуль статистика, занимается отображением различной статистики по проведенным занятиям и выполненным задачам студентами, например, сколько попыток принял студент, для того чтобы выполнить ту или иную задачу.

Модуль студент, отвечает за взаимодействия студентов с системой и состоит из двух небольших подмодулей занятие и статистика. Занятие позволяет контролировать ход занятия и смотреть различные материалы, предоставленные преподавателем для определенного занятия.

Главный модуль системы это IDE, он отвечает за выполнение задач. Текстовый редактор отвечает за ввод кода и проверки его синтаксиса. Модуль терминал, позволяет с помощью различных команд запускать, компилировать написанный код. Модуль задачи отвечает за отправку написанной программы на проверку и выдачу результатов после проверки кода.

3.2 Архитектура базы данных

3.2.1 Построение концептуальной модели данных

Данные веб – приложения необходимо систематизировать, хранить и обрабатывать, в качестве чего используется два источника данных:

– База данных

– Файловая система

Для того чтобы выбрать систему управления базой данных (СУБД) необходимо рассмотреть несколько предпосылок, которые определяют выбор

СУБД. Учитывая бизнес – правила, что будет хранится в базе, какие будут ограничения на операции я данными хранящимися в базе. Самыми часто используемыми операциями при использовании СУБД являются чтение данных и добавление. Так как некоторые данные имеют иерархическую структуру и эти данные могут отличатся друг от друга некоторыми атрибутами, в зависимости от типа, а также то что требования могут быть изменены, влача за собой возможное изменение атрибутов, было принято решение в СУБД использовать ту, в которой есть поддержка неструктурированной модели данных – NoSQL [6].

На основании предпосылок и бизнес – требований, было сделано заключение что будущая СУБД должна иметь:

- Поддержку NoSQL
- Возможность горизонтальной масштабируемости, в частности репликации БД
- Поддержка типов данных JSON, так как форматом данных для GraphQL является JSON

На основании всего выше изложенного было принято в качестве решения использовать СУБД MongoDB. MongoDB имеет интерфейс для выполнения запросов на языке JavaScript, возможности асинхронной репликации, поддерживает необходимые типы данных, которые могут понадобиться при проектировании БД согласно предметной области, например, массив, дата, объект и другие. Рационально подходит для хранения иерархических данных.

ER – диаграмма представлена на рисунке А.3 приложения А.

3.2.2 Модель данных СУБД

В качестве базы данных используется MongoDB. База данных в MongoDB состоит из коллекций. Каждая коллекция в базе, как правило соотносится с сущностью предметной области. Каждая коллекция состоит из документов, в которых хранятся записи в виде JSON – формата, с обязательным идентификатором ObjectId. ObjectId – является одним из представленных в MongoDB форматов данных. Если сравнивать с реляционными базами данных, то получается, что коллекция — это таблица, а документ — это запись в таблице(строка).

Ниже представлен список типов данных используемых в MongoDB:

- Integer – используется для хранения целочисленных значений. В зависимости от сервера может быть, как 32-битным, так и 64-битным.
- String – Используется для хранения символьных строк. В MongoDB используется кодировка UTF-8.
- Double – Используется для хранения значений с плавающей точкой.
- Boolean – Используется для хранения логических (true / false) значений.
- Arrays – Используется для хранения массивов значений по одному ключу.
- Object – Используется для встроенных документов.
- Null – Используется для хранения значения Null.
- Timestamp – Используется для хранения даты и времени.
- Object ID – Используется для хранения ID документа.
- Regular Expression – Используется для хранения регулярных выражений.
- Code - Используется для хранения JavaScript кода в документе.
- Date - Используется для хранения текущей даты или времени в UNIX формате.

Целостность связей в MongoDB не поддерживается, эта задача всегда решается на уровне кода сервера, что и было реализовано с использованием

GraphQL. Для того чтобы иметь полное представление отношений между сущностями была разработана модель данных СУБД на основе концептуальной модели данных изображенной на рисунке 3.2, на которой изображены псевдо-связи, рисунок 3.3.

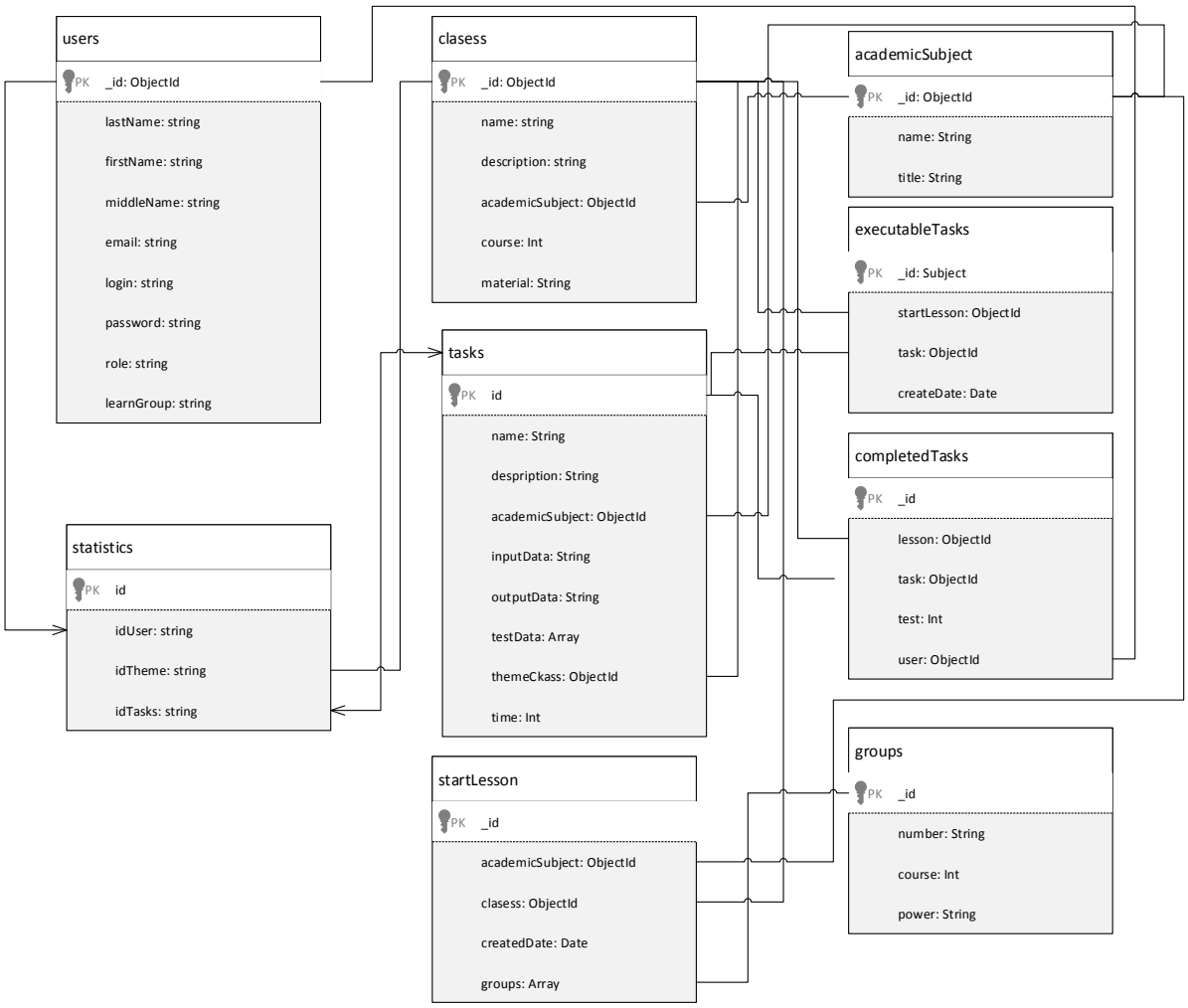


Рисунок 3.3 – Модель данных

В дополнении можно сказать, что наименование коллекций и псевдо-связи в MongoDB могут представлять совершенно иной смысл нежели таблицы в реляционных базах данных так коллекция может содержать вложенные объекты. При проектировании схем для базы MongoDB были использованы некоторые ее особенности:

– Ссылка – уникальный идентификатор ObjectId внешней сущности
– Вложение – полное содержание объекта одной сущности в другой.

При проектировании модели данных для базы данных MongoDB были определены особенности проектирования:

- При составных псевдоключах резко теряется гибкость работы сданными.
- Хранение только идентификатора объекта внешней сущности потребует выполнить дополнительный запрос для получения всех данных внешней сущности.
- Вложение объекта сущности имеет смысл тогда, когда вложенный объект будет встречаться вместе с родительским.
- Ссылки обеспечивают большую гибкость при частых изменениях вложенной сущности.

3.3 Построение диаграммы классов

Для отображения статической структуры модели систем необходимо построить диаграмму классов. Диаграмма классов [] языка UML представляет собой граф, на котором представлен совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающими их отношениями. Диаграмма классов может содержать интерфейсы, пакеты, отношения, отдельные экземпляры классификаторов, такие как объекты и связи.

Для реализации данной системы, была построена объектная модель, представленная на рисунке 3.4, которую после заполнения необходимо будет развернуть в программный код. Объектно-ориентированная технология означает построение программы как набора взаимодействующих и независимых объектов (классов), представленных экземплярами абстрактных типов, данных и обрабатывающих информацию посредством передачи сообщений друг другу. Объединяя в единое целое данные и процедуры, классы позволяют унифицировать обращение к различным типам данных [1].

На основе анализа предметной области, представленного в разделах 1 и 2, и построенной объектной, можно выделить следующие необходимые классы для нашей системы:

1. Класс User (рисунок 3.5). Пользователи системы. Исследование предметной области выявило, что имеется два типа пользователей, различающихся возможностями работы в этой системе:

– Преподаватель

– Студент

Пользователь обладает функционалом: обновление, создание, просмотр статистики, просмотр списка пользователей, выполнение заданий, создание заданий.

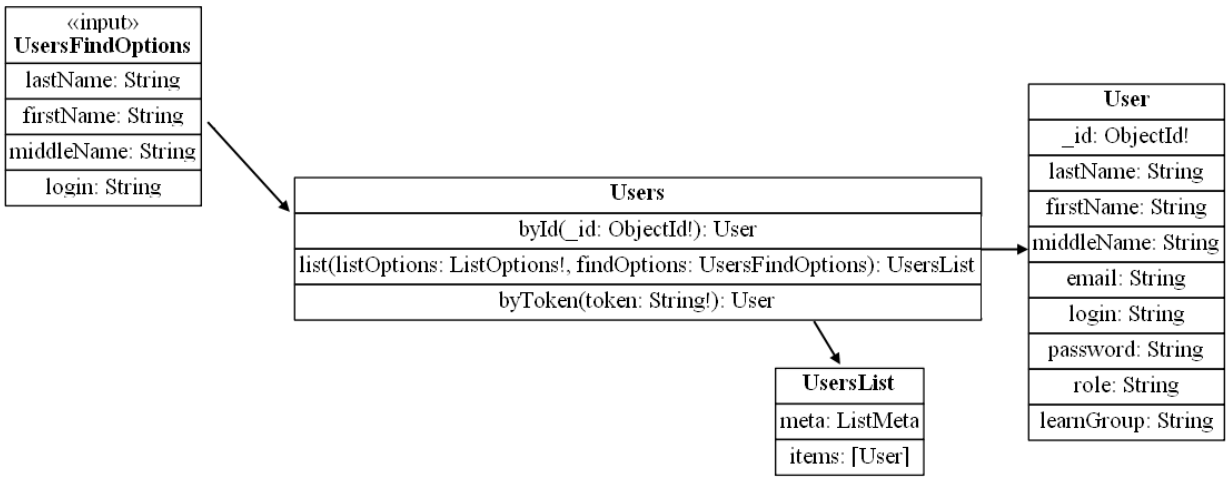


Рисунок 3.5 – Диаграмма класса User

2. Класс Class. Занятия

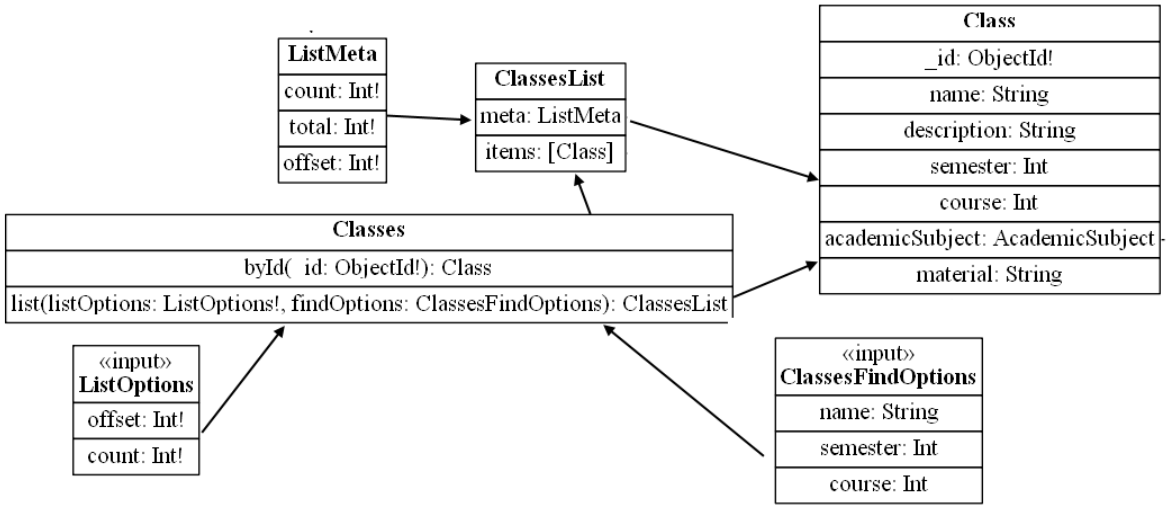


Рисунок 3.6 – Диаграмма класса Class

3. Класс Task. Задачи

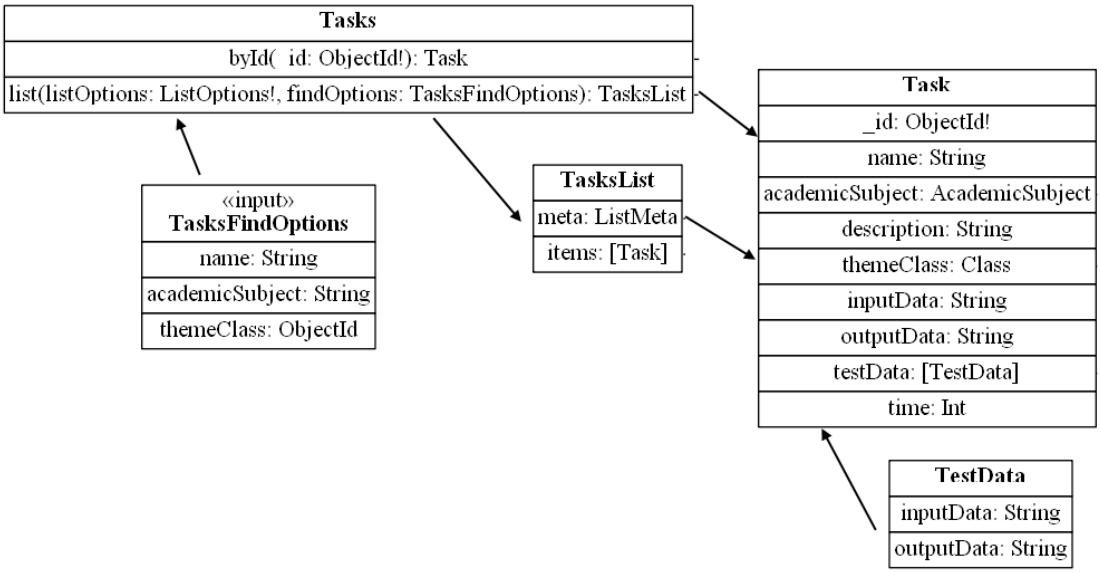


Рисунок 3.7 – Диаграмма класса Task

4. Класс CompletedTask – Выполнение задачи.

Данный класс отвечает за работу с данными завершенного задания, он имеет несколько полей, наследуемых от других классов, например, от класса User.

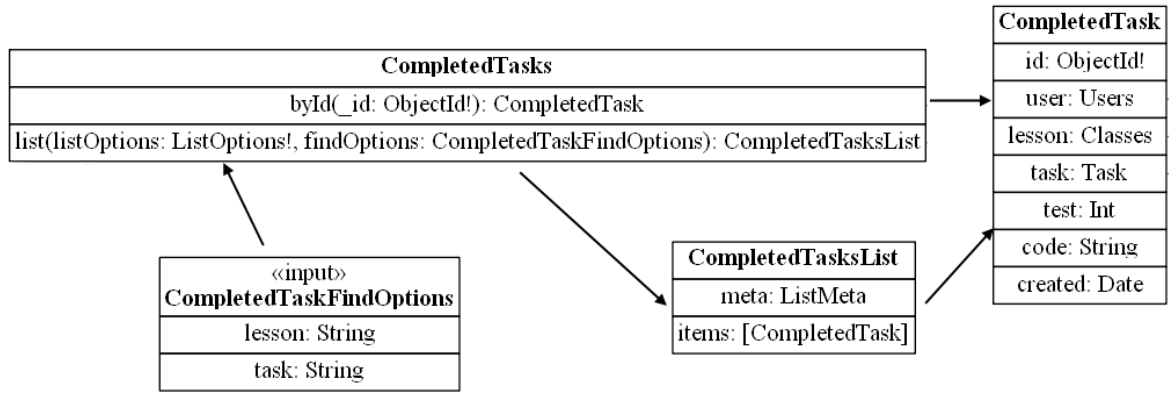


Рисунок 3.8 – Диаграмма класса CompletedTask

3.4 Программная реализация системы

3.4.1 Клиентская часть

Клиентская часть приложения отвечает за работу веб – приложения в браузере клиента. Для оптимального удовлетворения бизнес – требований было принято решение разрабатывать систему с использованием технологий SPA (Single Page Application).

SPA – это приложение, которое поставляется в браузер, который не требует перезагрузки страницы во время использования. Как и все приложения, он призван помочь пользователю выполнить задачу, например, «написать документ» или «администрировать веб-сайт». Мы можем думать о SPA как о живом клиенте, который загружается с веб-сервера [2].

Особенностью SPA является, то что все промежуточные действия для получения данных происходят на клиенте и только данные принципиальных

действий отправляются на сервер, где в ответ приходят необходимые данные для продолжения работы. Это помогает уменьшить зависимость от качества связи между сервером и клиентом. Без запросов клиента на сервер, может происходить смена представлений пользовательского интерфейса согласно UX/UI [3]. В случае если все-таки требуется обмен данными с сервером, запросы выполняются асинхронно, что позволяет продолжить работу не дожидаясь ответа от сервера. За счет того, что некая часть бизнес-логики переносится на клиент и происходит отделения данных от UI, достигается слабая связь между клиентом и сервером. Сравнивая SPA и традиционный подход к организации веб – приложений, работа клиентской части SPA – приложений характеризуется высоким быстродействием и возможностью корректной работы при плохом соединении с Интернетом.

Клиентское приложение, это приложение которое выполняется при помощи браузера, изменяя или расширяя его возможности, организуя уровень абстракции для работы с бизнес – логикой. Для такой организации клиента часто используют браузерные технологии такие как Silverlight, Flash, JavaScript.

В данном случае, в качестве средства реализации клиентской части приложения выбор пал на язык JavaScript.

Для достижения более гибкой и богатой функциональности клиентской части приложения, целесообразно использовать фреймворк. В моем случае, было принято использовать библиотеку React [4].

В качестве способа организации обмена данными с API, оптимальным решением было использовать технологию GraphQL. GraphQL — это стандарт декларирования структуры данных и способов получения данных, который выступает дополнительным слоем между клиентом и сервером [5]. Плюсы GraphQL:

– GraphQL облегчает агрегацию данных из нескольких источников.

– GraphQL использует систему типов для описания данных.

– GraphQL не зависит от протокола передачи данных, может использовать любой (http, ws, ssh, cli, etc.)

– В GraphQL для работы с данными мы всегда обращаемся к единой точке входа — GraphQL серверу. Изменяя структуру, поля, параметры запроса мы работаем с разными данными.

– GraphQL дает возможность написания документации непосредственно в коде (inline documentation).

– Формат и структура данных определяется на стороне клиента.

Для обмена данными между клиентом и сервером в качестве основного средства было принято использовать протокол HTTP. В некоторых случаях будет использоваться WebSockets.

3.4.2 Серверная часть

Основной целью приложения является предоставление услуг по средствам сети Интернет, в серверную часть необходимо включить веб – сервер, предназначенный для обслуживания HTTP – запросов.

Так как приложение предполагает большое число операций по чтению, записи и редактированию больших объемов данных, наиболее удобным вариантом было включить в серверную часть технологии баз данных.

Серверная часть включает в себя веб – сервер и сервер с СУБД. В задачи веб – сервера включены:

– Получение и отправка ответов на HTTP – запросы

– Предоставление приложению доступа к необходимым модулям

– Авторизация и аутентификация пользователей

– Реализация функций файл-сервера

В качестве основного языка разработки серверной части был выбран JavaScript. Некоторые из причин сделанного выбора:

– MongoDB имеет интерфейс для выполнения запросов на JavaScript

– JSON нативный формат JavaScript и используется для обмена данными между клиентской частью и API серверной части

– Использование одного языка JavaScript и на клиенте, и на сервере, и при работе с СУБД, позволит упростить и ускорить процесс разработки, использовать одни и те же библиотеки на всех уровнях веб приложения

Для того чтобы выполнять код написанный на JavaScript на сервере, он должен быть интерпретирован. В качестве платформы для выполнения JavaScript кода был выбран Node.JS, так как:

– Работает согласно асинхронной модели

– Имеет в наличии часто используемые библиотеки и модули, написанные на JavaScript

В качестве базового фреймворка для создания http – сервера на Node.JS будет использоваться ExpressJS. Основные возможности express:

– Гибкая система маршрутизации запросов

– Динамические представления

– Перенаправления

– Обработка представлений и поддержка частичных шаблонов

– Поддержка конфигураций на основе окружений

– Оповещения, интегрированные с сессиями

– Максимальное покрытие тестами

– Утилиты для быстрой генерации основы приложений

– Настройки представлений на уровне приложений

4 Функциональность системы

4.1 Список занятий

Отображение списка (рисунок 4.1) производится по запросу «/teacher/classes», после выполнения запроса занятия отображаются в виде таблицы, в которой указаны их названия, предметы по которым эти занятия проводятся и другие дополнительные данные, которые подгружаются из БД. Отображение списка занятий доступны только преподавателям.

На этом экране также присутствуют кнопки просмотр и редактирование занятия, расположенные напротив каждого элемента списка. Внизу списка расположена кнопка добавления нового занятия.

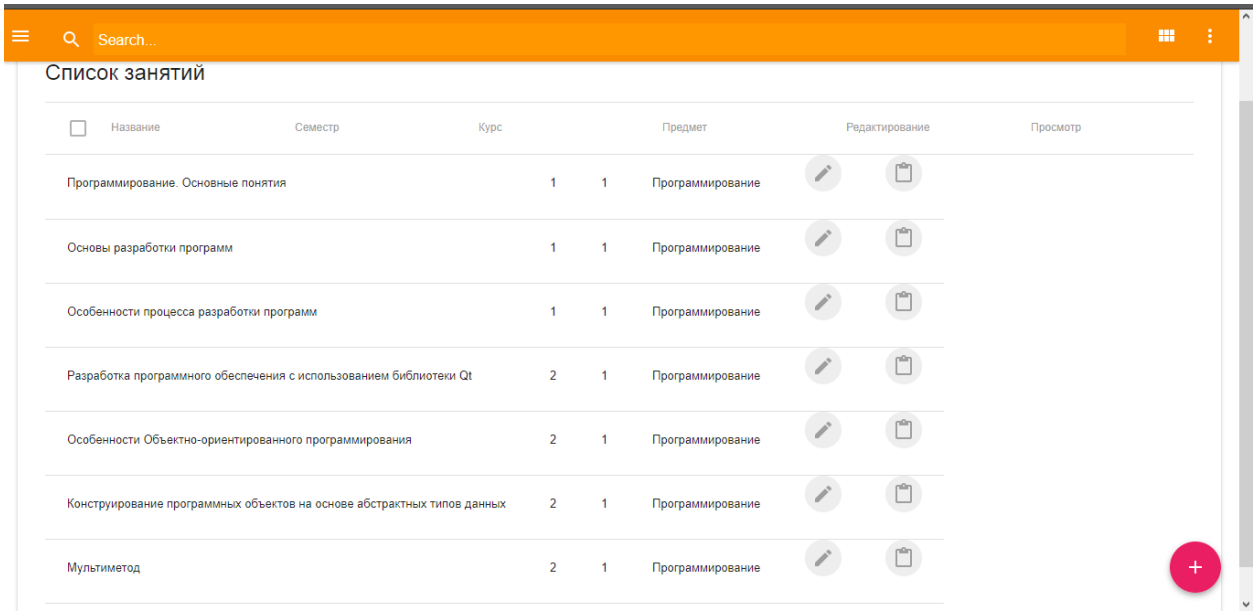


Рисунок 4.1 – Отображение списка занятий

4.2 Добавление / редактирование занятий

В данном окне, расположенном на рисунке 4.2 производится добавление редактирование информации о занятии. Вводится тема занятия, короткое описание занятия, добавляется курс для которого он рассчитан, например, первый или второй, из списка выбирается предмет к которому относится

данное занятие. И в конце добавляется информация для занятия которое будет демонстрироваться во время проведения этого занятия, например, в виде презентации.

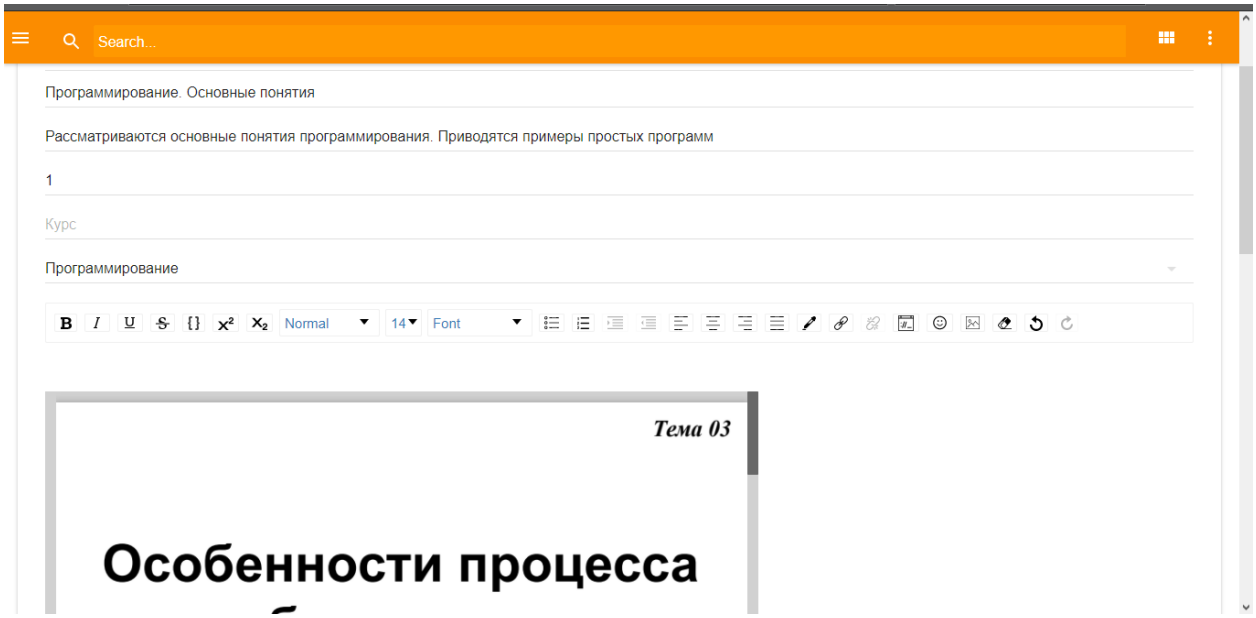


Рисунок 4.2 – Добавление / редактирование занятия

4.3 Добавление / редактирование задачи

Данный функционал предназначен для работы с задачами, вносится вся информация по задаче которую потом будут решать студенты. Вводимая информация, это название, условие задачи, предмет и тема занятия выбираются из списка сформированного из ранее добавленной информации по занятию, в конце вводится входные, выходные данные, для того чтобы студенты могли убедится, что задача решена верна и также добавляются тестовые данные для проверки правильности выполнения задачи, эти тестовые данные нужны системе проверки, в которую студент отправляют выполненные задачи.

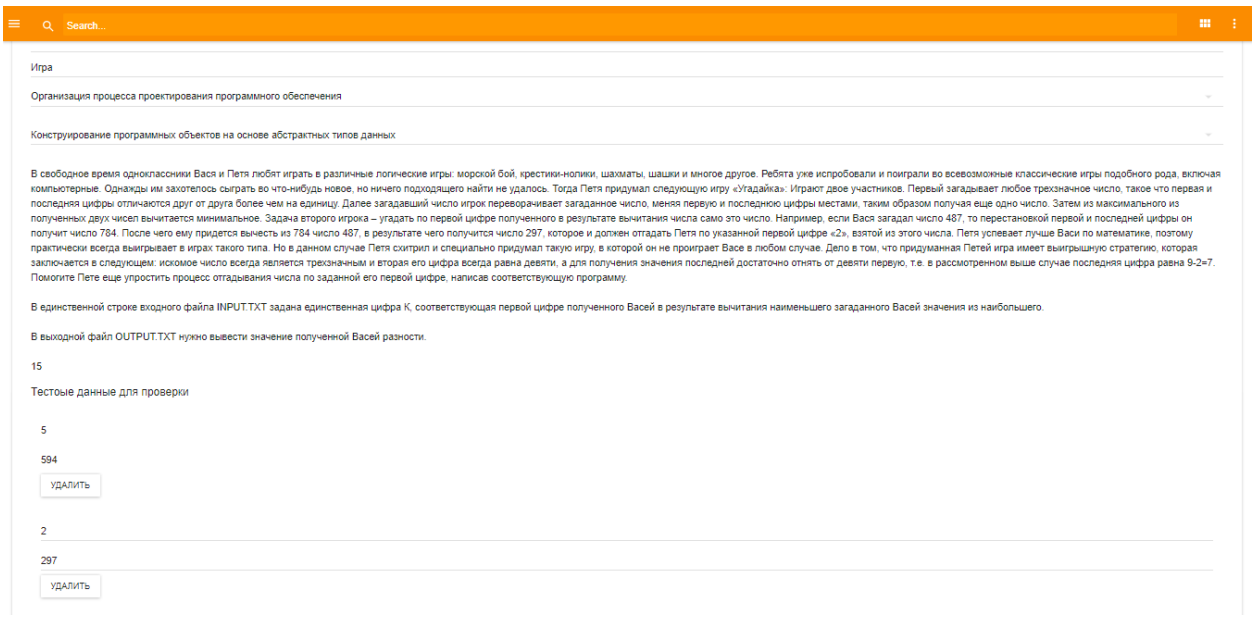


Рисунок 4.3 – Добавление / редактирование задачи

4.4 Проведение занятия

Данная функция состоит из нескольких этапов. На начальном этапе (рисунок 4.4) преподаватель выбирает группы у которых будет происходить занятие в данный момент. После выбора групп, происходит выбор предмета для занятия (рисунок 4.5). Выбор темы (рисунок 4.6) происходит в зависимости от выбранного предмета, для каждого предмета доступны свои темы. После выбора темы занятия открывается окно с информацией о занятии (рисунок 4.7) в котором доступна презентация, если ее добавляли в занятие, данную презентацию преподаватель если захочет может демонстрировать студентам при помощи проектора. Такая же презентация будет доступна всем студентам у которых проходит данное занятие, студент может посмотреть презентацию при входе в систему под своим аккаунтом. После прочтения лекции или делая паузы в занятии, преподаватель может давать студентам задачи путем их выбора из доступных (рисунок 4.8). После выбора задания преподавателю открывается окно с информацией о задаче (рисунок 4.9), где преподаватель может настроить задачу, до того, как ее начнут студенты, например, преподаватель может настроить время за которое студенты должны ее

выполнить. После настройки задачи, преподаватель запускает выполнение и в этот момент студентам становится доступно выполнение задач, а преподавателю стекается вся информация в режиме реального времени о ходе выполнения студентами (рисунок 4.10)

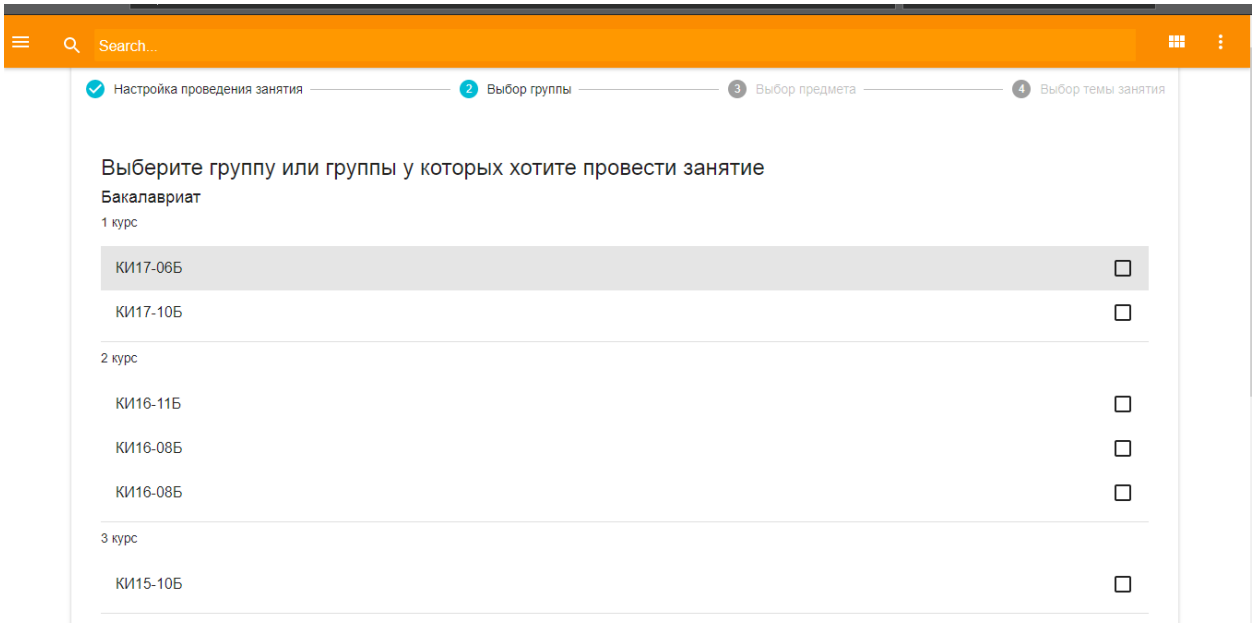


Рисунок 4.4 – Выбор групп

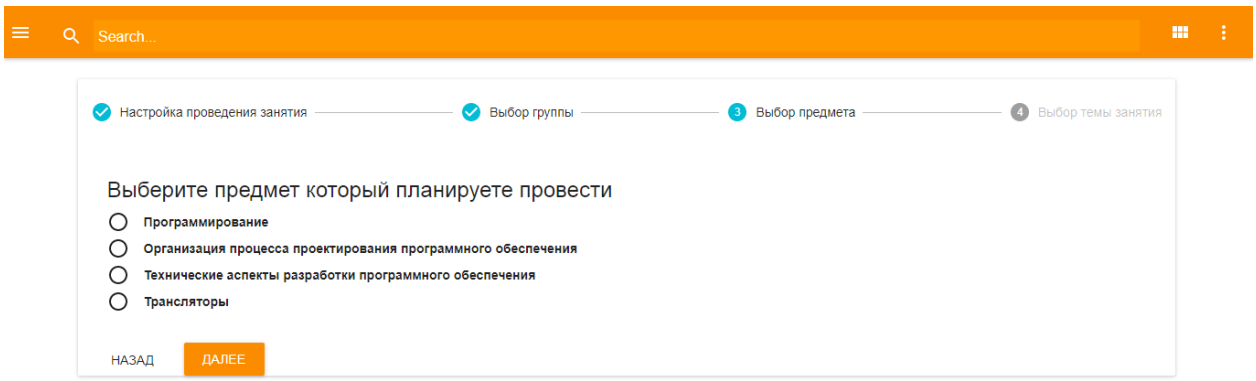


Рисунок 4.5 – Выбор предмета

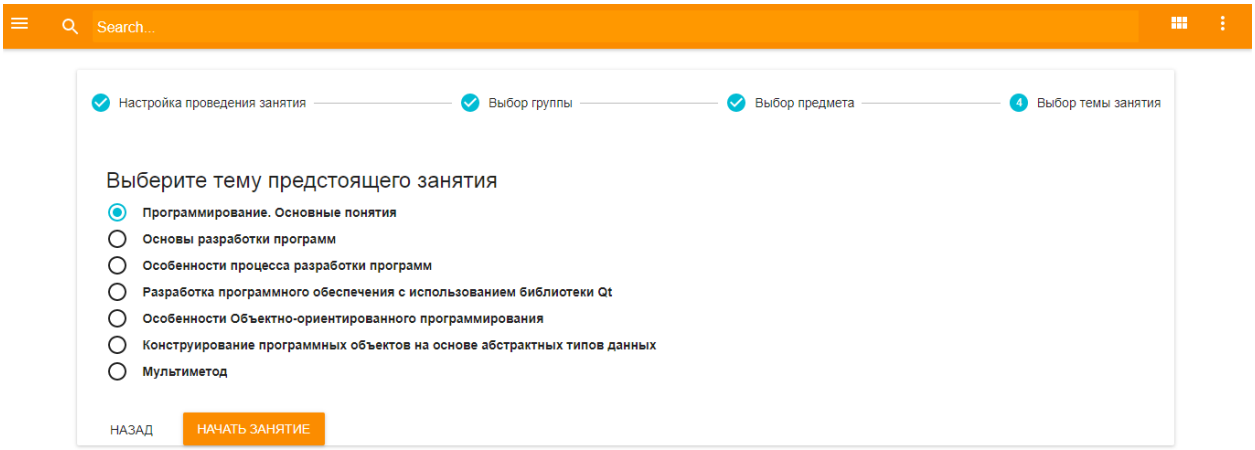
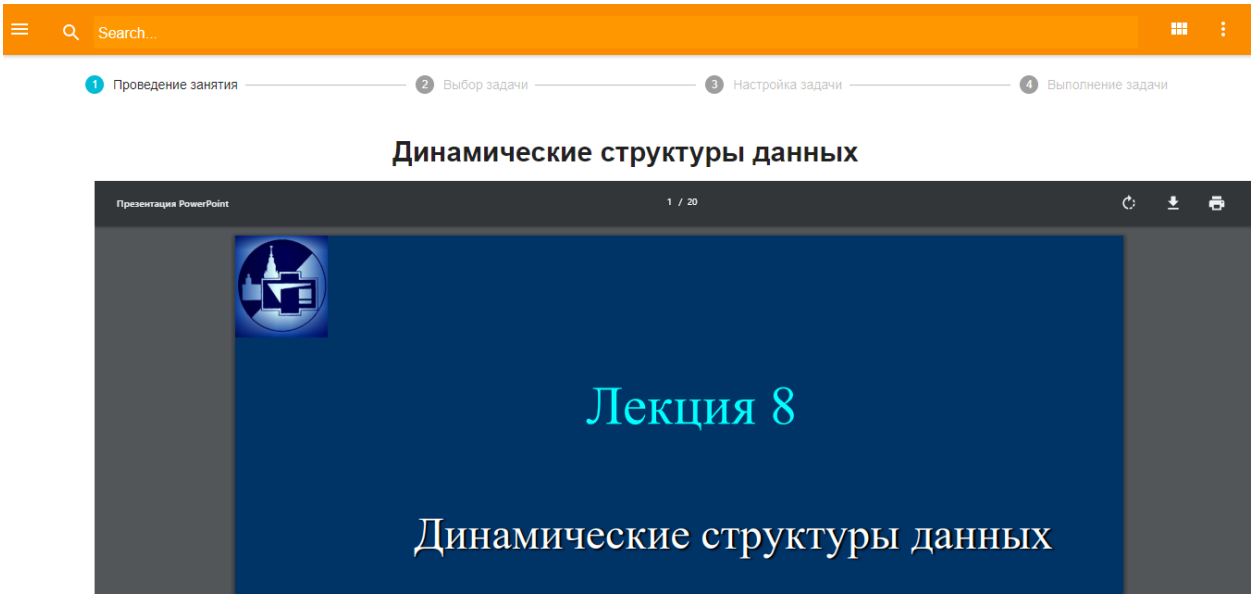


Рисунок 4.6 – Выбор темы



4.7 – Отображение лекции (презентации)

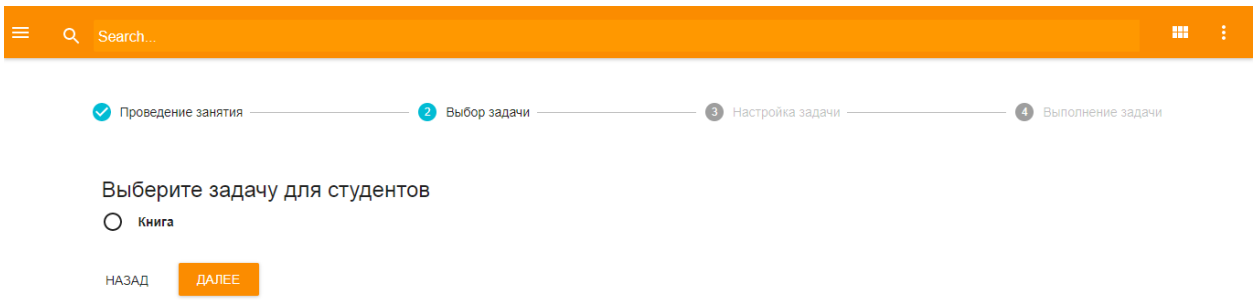


Рисунок 4.8 – Выбор задачи

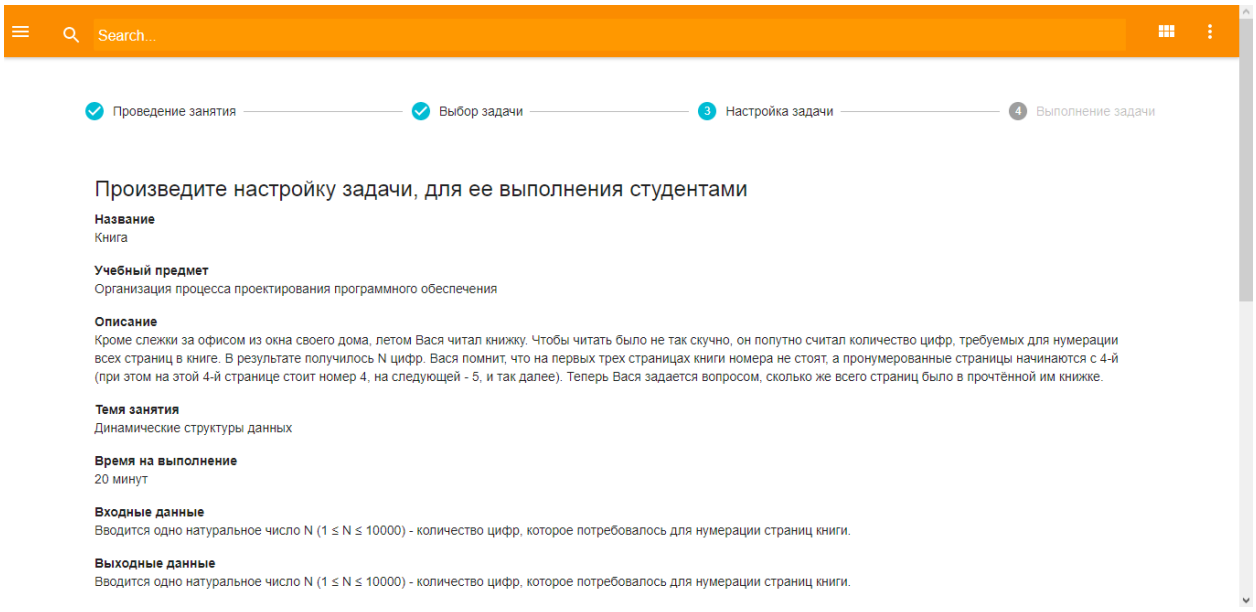


Рисунок 4.9 – Настройка задачи

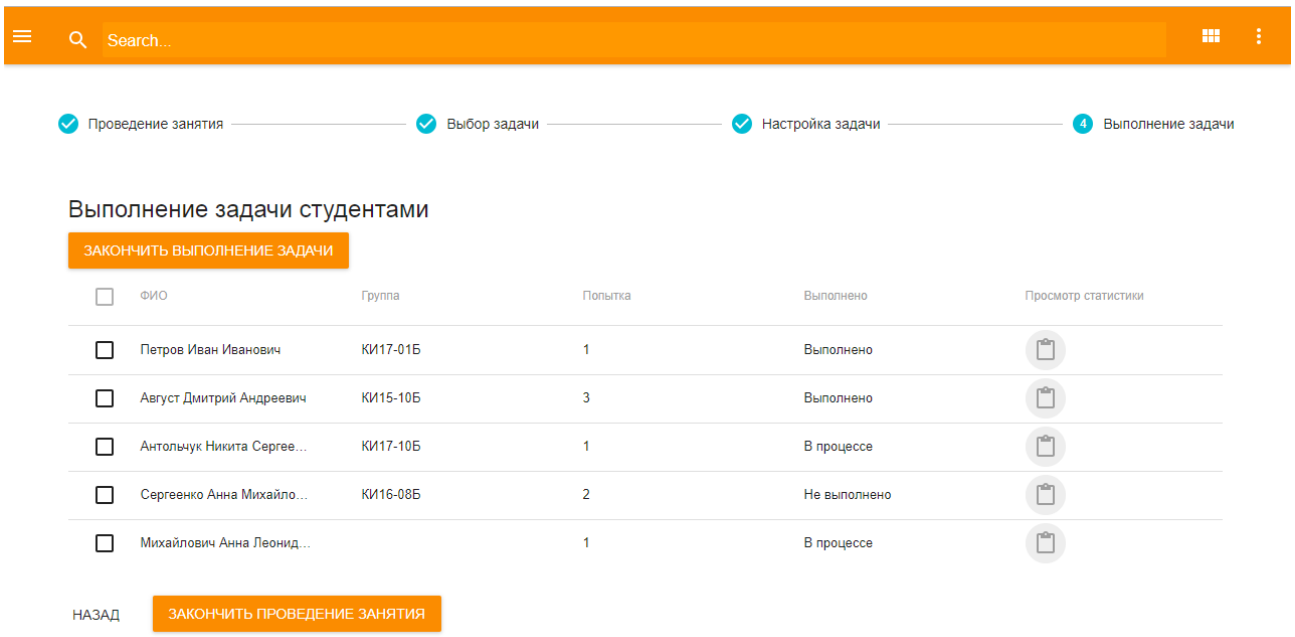


Рисунок 4.10 – Выполнение заданий

4.5 Просмотр статистики

Для преподавателя доступно несколько видов статистики, первый статистика по одному студенту (Рисунок 4.11), предназначена для просмотра успеваемости студента, то как он справляется с выполнением выданных ему задач, второй вид статистики общая статистика (Рисунок 4.12) по занятию.

Индивидуальная статистика показывает, какие задачи выполнял студент, какие у него успехи по выполненной задаче и с какой попытки.

Общая статистика показывает количество студентов, которые находятся на лекции студентов, можно увидеть кто из студентов лучше справился, а кто вообще не преступал к выполнению.

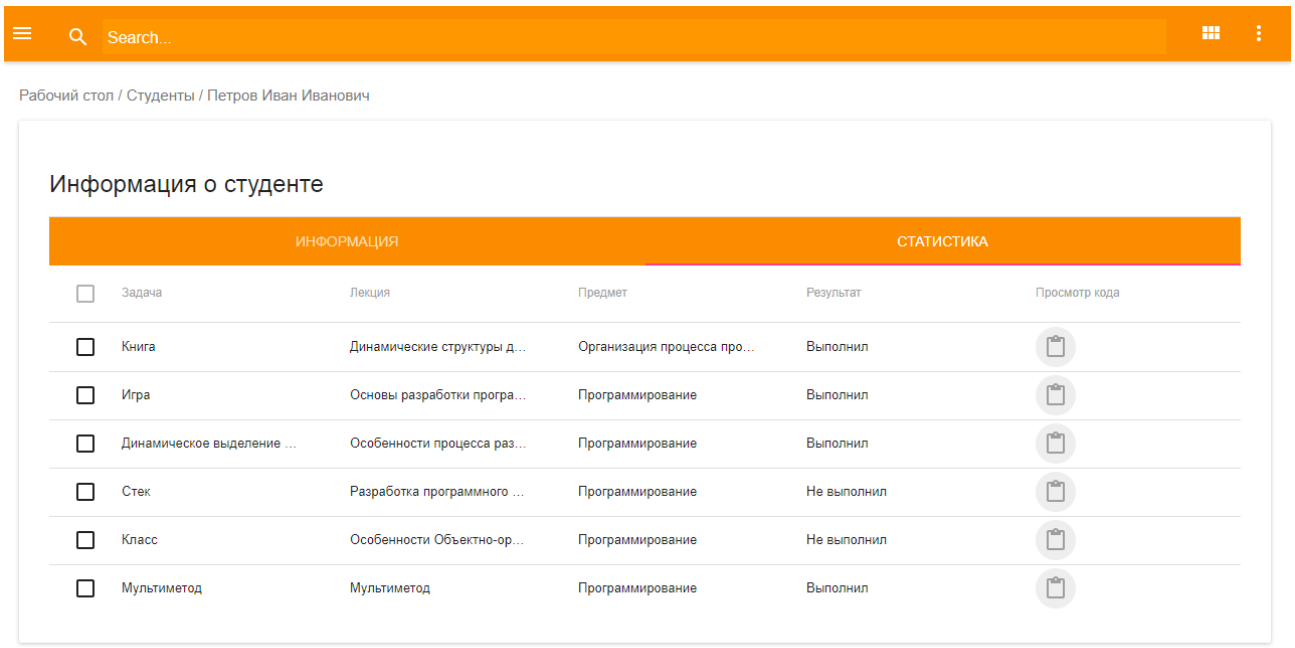


Рисунок 4.11 – Статистика студента

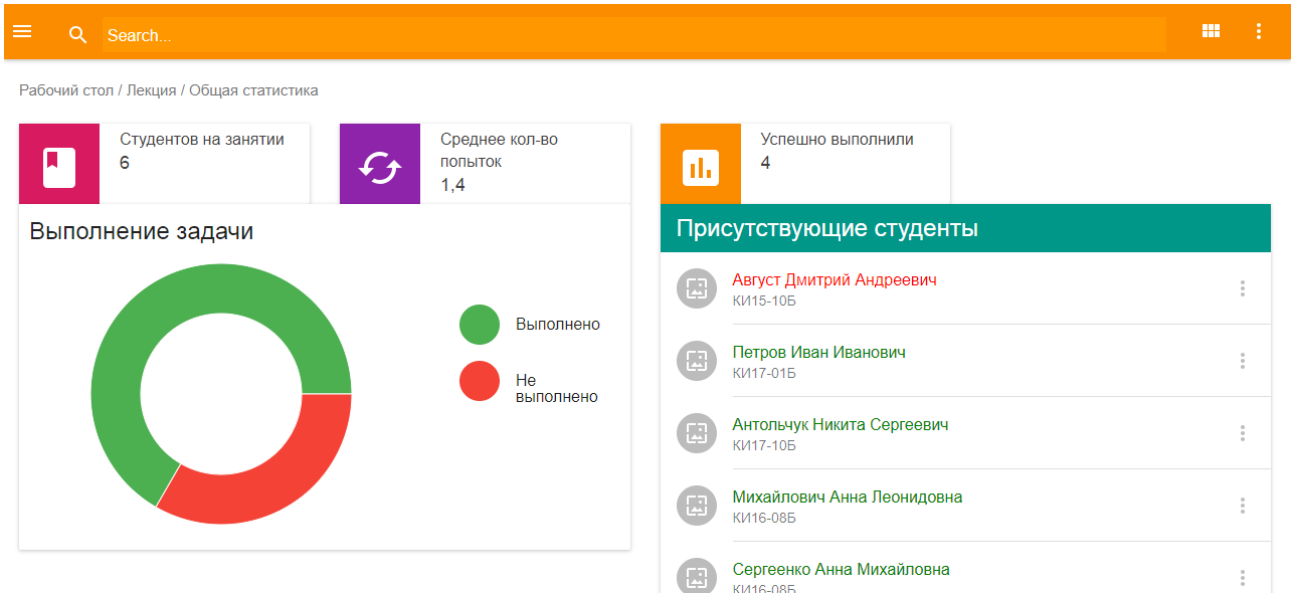


Рисунок 4.12 – Общая статистика

4.6 IDE

IDE доступно как для студента так и для преподавателя. Студент открывает IDE во время лекции либо может самостоятельно открывать и писать код во вне учебное время. Во время занятия и открытой задачи студент пишет код и отправляет его на проверку по средством нажатия пункта меню Task ->

Review. После проверки студенту приходит сообщение о том на сколько он правильно решил задачу. Во время решения задачи студент может посматривать условия задачи. После того как он ее решил он ее запускает, запускать можно либо с помощью командной строки доступной внизу экрана, или с помощью меню.

Преподаватель в IDE может смотреть код студентов и каким-то образом его модифицировать и сохранять, для того чтобы студент потом увидел замечания и комментарии которые ему оставил преподаватель.

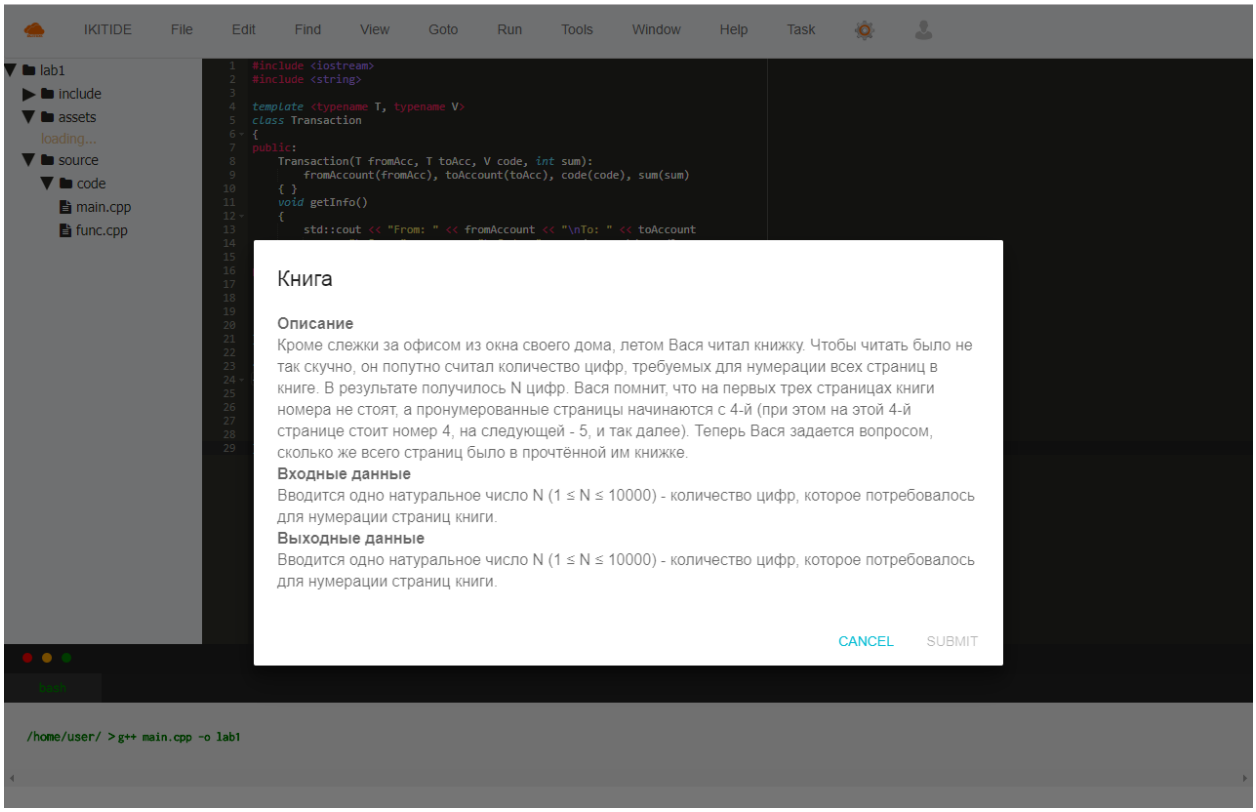


Рисунок 4.13 – Экран IDE

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы над магистерской диссертацией, были выполнены следующие задачи.

1. Произведен анализ особенностей процесса обучения для определения основных подходов к повышению уровню интерактивного взаимодействия между преподавателем и студентами во время проведения лекционных занятий, сформированы рекомендации по повышению интерактивности.

2. Разработана клиент – серверная архитектура приложения. Была разработана база данных, которая позволяет хранить данные пользователей, управлять задачами, и все что связано с выполнением задач, т.е. лекциями и тестами.

3. Был разработан прототип системы, который обладает функционалом, позволяющим использовать его во время проведения лекций. Данный прототип демонстрирует возможность проведения лекций в режиме интерактивности, что позволяет в любой момент остановить процесс подачи теоритического материала, и выдать небольшую задачу, для самостоятельного решения студентами.

4. Было произведено тестирование всех основных функций прототипа системы, таких как выдача задач, получение статистики, выполнение заданий студентами и др.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Калашников, В. И. и др. Информационно-измерительная техника и технологии / В. И. Калашников, С. В. Нефедов, А. Б. Путилин. – Москва: Высшая школа. – 2002. – 520 с.

2. Mikowski M. S., Powell J. C. Single page web applications //B and W. – 2013.

3. Park R. C. et al. Picocell based telemedicine health service for human UX/UI //Multimedia Tools and Applications. – 2015. – Т. 74. – №. 7. – С. 2519-2534.

4. React - A JavaScript library for building user interfaces URL: https://reactjs.org (дата обращения: 08.02.2018).

5. GraphQL | A query language for your API URL: http://graphql.org (дата обращения: 15.01.2018).

6. Freeman S. et al. Active learning increases student performance in science, engineering, and mathematics //Proceedings of the National Academy of Sciences. – 2014. – Т. 111. – №. 23. – С. 8410-8415.

7. Заболотнева, О. Л. Специфика лекции в образовательном пространстве вуза //Слово, высказывание, текст в когнитивном, прагматическом и культурологическом аспектах. – 2014. – С. 182-184.

8. Акулич, М. М. Образование в условиях глобализации [Текст] / М. М. Акулич // Университетское управление. – 2005.— № 5 (38).

9. Dolnicar, S. (2005). Should we still lecture or just post examination questions on the web?: The nature of the shift towards pragmatism in undergraduate lecture attendance. Quality in Higher Education, 11(2), 103-115. http://ro.uow.edu.au/commpapers/299/

10. Cloud9 [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Cloud9

11. Cloud9 – Your development environment, in the cloud [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/cloud9/?origin=c9io

12. Codeboard – the IDE for the classroom [Электронный ресурс] – Режим доступа: https://codeboard.io

13. CS50 Docs [Электронный ресурс]. – Режим доступа: https://manual.cs50.net/cs50-ide/online.html

ПРИЛОЖЕНИЕ А

Диаграммы системы

В приложении приводятся диаграммы разработанные в процессе проектирования архитектуры системы.

На рисунке А.1 представлена диаграмма прецедентов пользователей, которая показывает взаимоотношения между пользователями и частью функциональности системы.

Рисунок А.2 – диаграмма деятельности преподавателя, данная диаграмма показывает возможные действия преподавателя в системе.

Рисунок А.3 ER – модель демонстрирует ключевые сущности системы и показывает связи между ними.

На рисунках А.4 и А.5 диаграммы классов, демонстрирует классы системы, основной класс данной диаграммы Query которые наследует все остальные классы системы. Класс Query взаимодействует с классом Mutation через класс Schema.

Диаграмма прецедентов пользователя и системы (рисунок А.6) – показывает взаимоотношения между пользователями и самой системой между собой.

Рисунок А.1 – Диаграмма прецедентов, описывающая основные варианты использования пользователями разрабатываемой системы

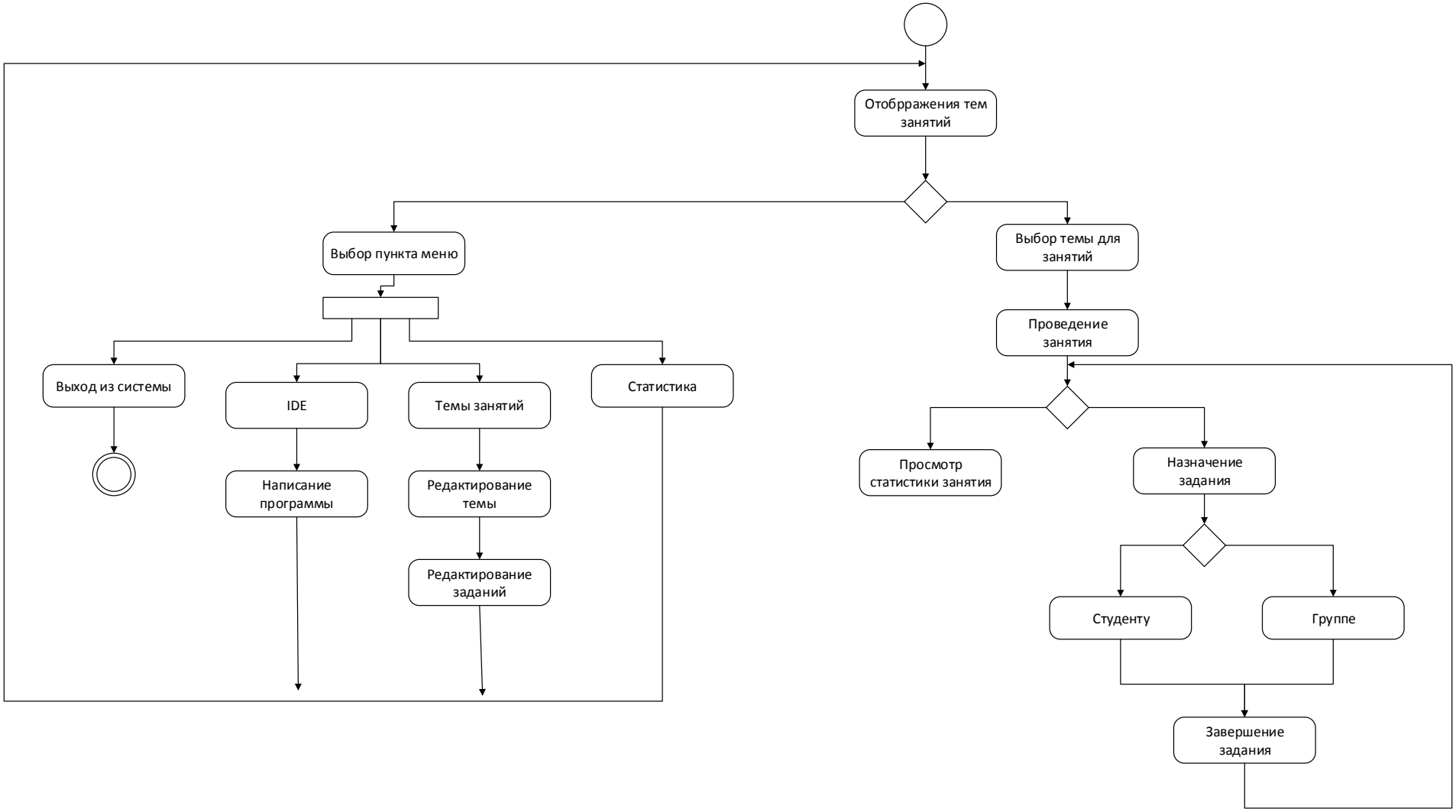


Рисунок А.2 - Диаграмма деятельности преподавателя

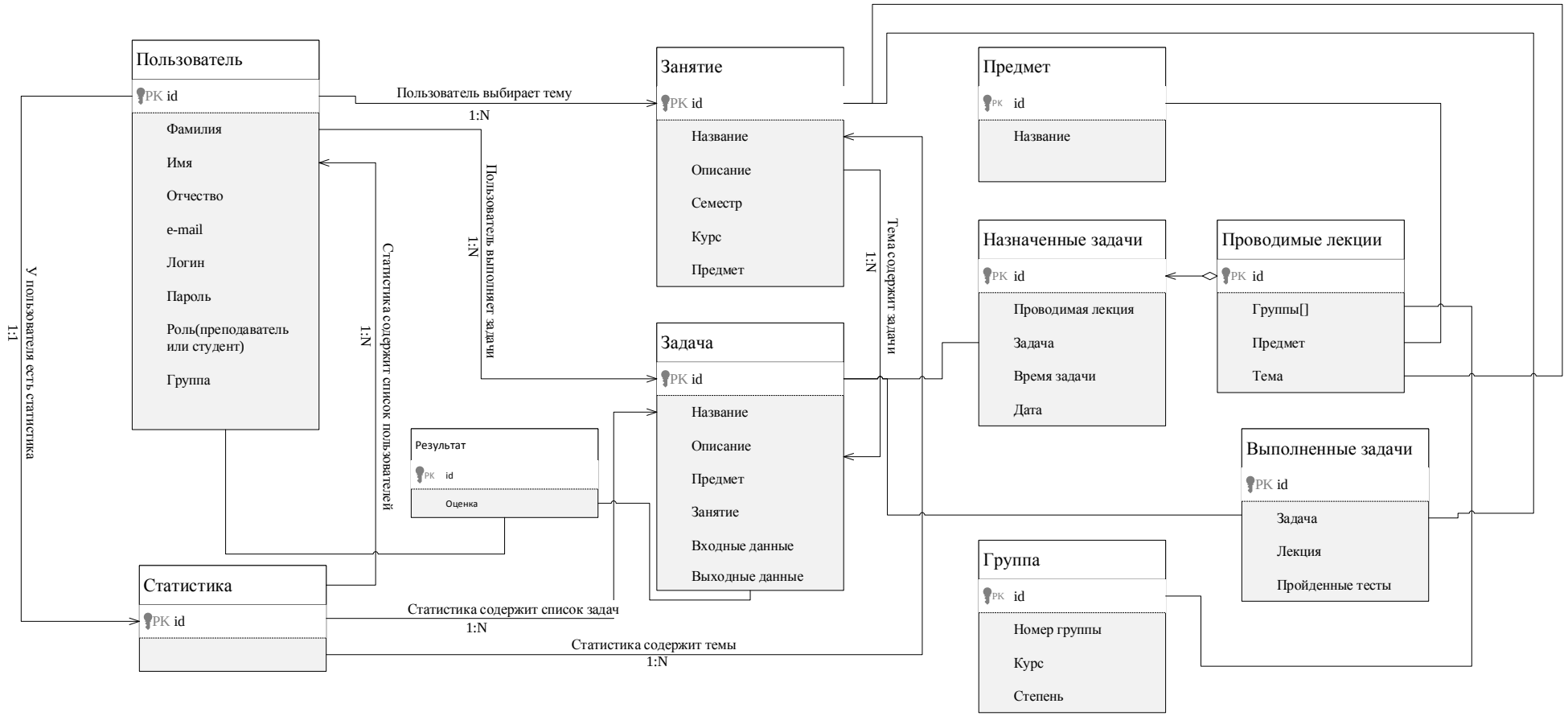


Рисунок А.3 – Модель данных, описывающая концептуальные схемы предметной области

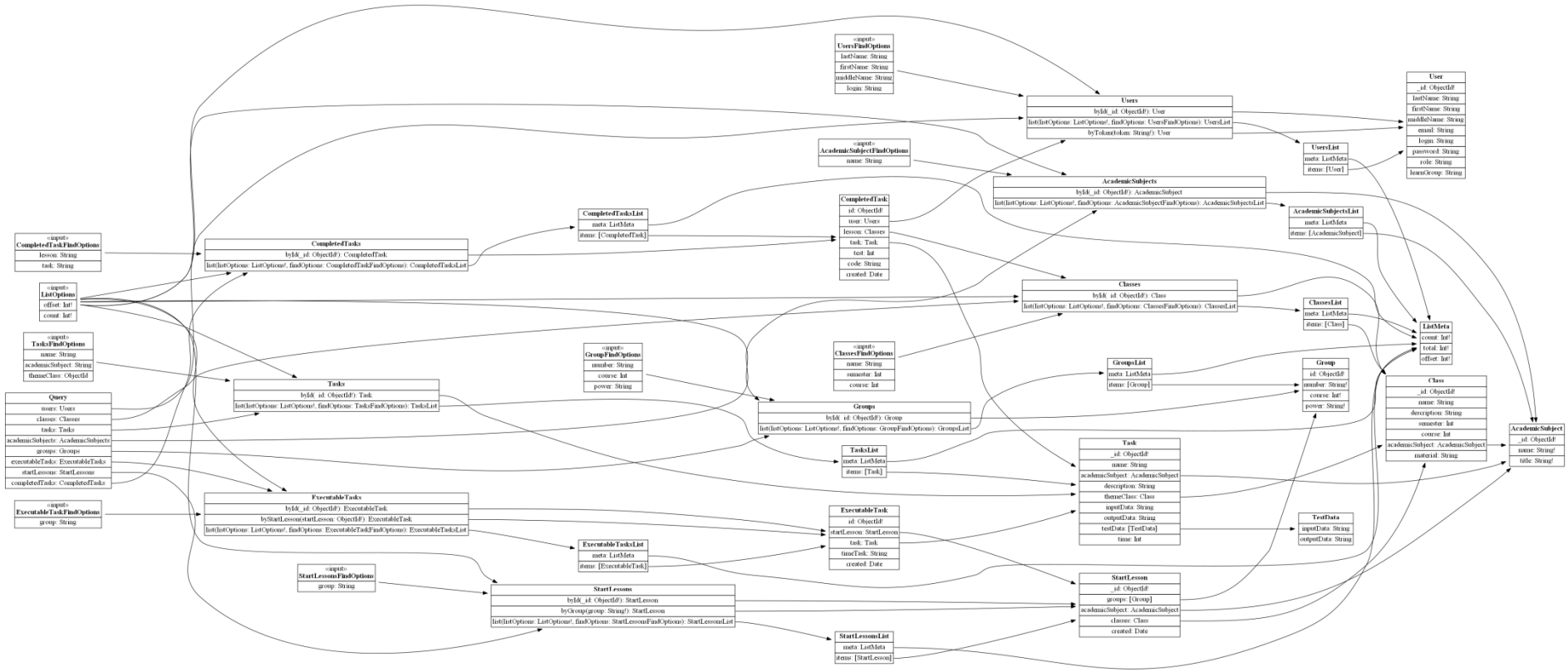


Рисунок А.4 – Диаграмма класса Query, обеспечивающая взаимодействие между базой данных и остальными классами

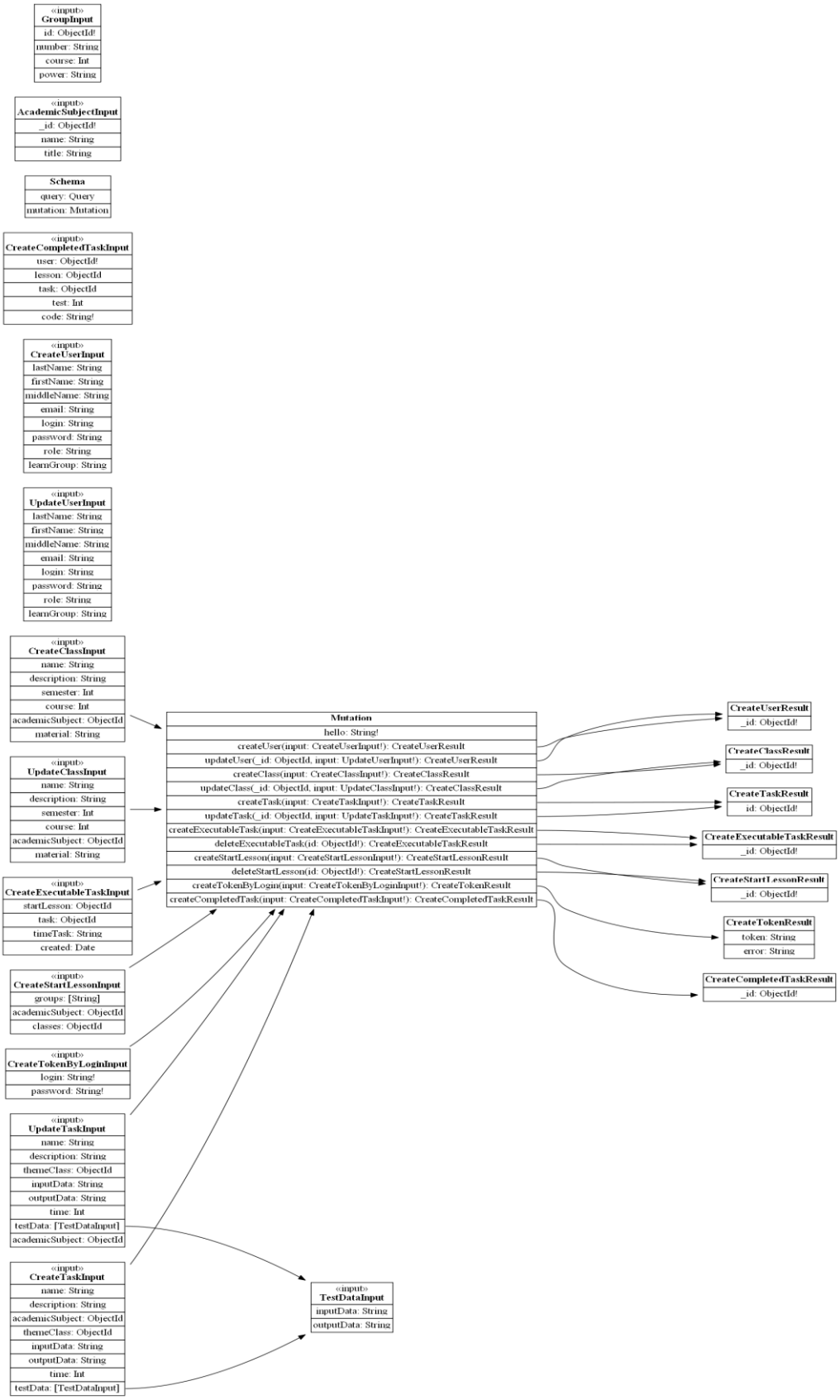


Рисунок А.5 – Диаграмма класса Mutation, обеспечивающий изменение информации в базе данных

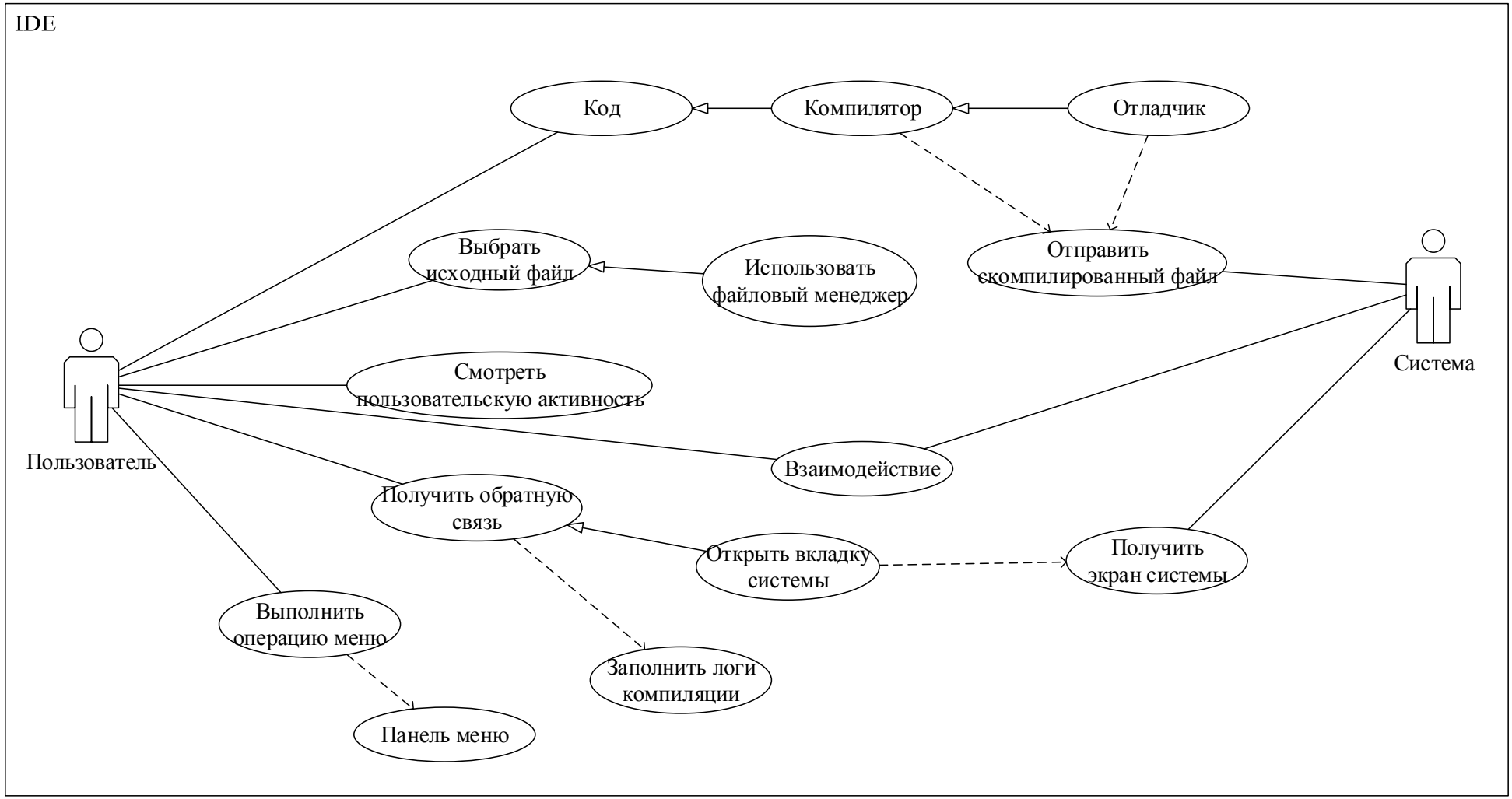


Диаграмма А.6 – Диаграмма прецедентов пользователя и системы, показывающая взаимодействие между друг другом